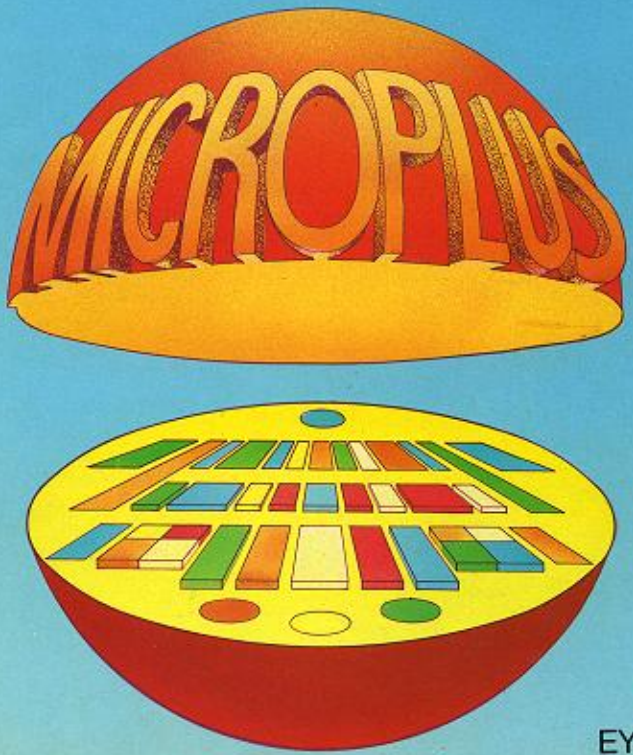
ALLER PLUS LOIN EN BASIC TO7



EYROLLES

JEAN-CLAUDE WANNER

ALLER PLUS LOIN EN BASIC T07

CHEZ LE MÊME ÉDITEUR

Dans la même collection

VANRYB-POLITIS	— *Tout savoir sur New-Brain*, 104 p. ; 1984.
GOLDSTEIN	— *Le guide de l'IBM-PC*, 272 p. ; 1984.
BUNN	— *Obtenez le maximum de votre ATARI*, (à paraître).
GARCIA	— *Le Besic minimum*. 15 mots pour apprendre à programmer, 1984, (à paraître).
WANNER	— *Aller plus loin en Besic TO7*, 312 p. ; 1984.
VANRYB-POLITIS	— *Tout savoir sur Multitech*, (à paraître).
POLITIS-VANRYB	— *Tout savoir sur ORIC*, (à paraître).

« La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article 40) ».

« Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal ».

© Éditions EYROLLES, 1984

ALLER PLUS LOIN EN BASIC TO7

par

Jean-Claude WANNER

Ancien élève de l'Ecole Polytechnique et de l'Ecole Nationale Supérieure de l'Aéronautique et de l'Espace

EYROLLES 61, boulevard Saint-Germain — 75005 Paris
1984

Si vous désirez être tenu au courant de nos publications, il vous suffit d'adresser votre carte de visite au :

Service «Presse», Éditions EYROLLES
61, Boulevard Saint-Germain,
75240 PARIS CEDEX 05,

en précisant les domaines qui vous intéressent. Vous recevrez régulièrement un avis de parution des nouveautés en vente chez votre libraire habituel

Avertissement

Avant de préciser les objectifs de ce manuel, annonçons tout de suite ce qu'il ne saurait être!

Ce n'est pas un cours d'informatique; l'informatique constitue l'ensemble des sciences et techniques qui permettent aux ingénieurs de concevoir des ordinateurs. Notre but n'est pas de délivrer un tel enseignement. Nous cherchons à utiliser au mieux le TO 7, non à le concevoir (c'est déjà fait!).

Ce n'est pas non plus un cours de programmation. Il est nécessaire pour le lire avec profit, de connaître quelque peu le langage Basic et de posséder certaines notions comme les boucles (FOR, NEXT), les sous-programmes (GOSUB), la lecture des données (DATA, READ), les instructions d'entrée-sortie (INPUT, PRINT), etc..

Le but de ce manuel est de présenter, par des exemples concrets, l'utilisation de la panoplie des instructions BASIC disponibles sur le micro-ordinnateur TO7.

Le manuel d'Initiation BASIC TO7 permet d'acquérir les notions de base et de les appliquer en réalisant des programmes simples.

Le manuel de Référence BASIC TO7 donne les règles d'utilisation de toutes les instructions BASIC disponibles sur le TO7. C'est un ouvrage fort utile au programmeur confirmé qui peut y trouver tous les renseignements nécessaires à une utilisation correcte de chaque instruction.

Mais ce n'est pas un manuel que l'on peut lire, de la première à la dernière page, pour apprendre à connaitre les différentes instructions; c'est un manuel que l'on consulte pour y "pêcher" le renseignement utile.

Entre le Manuel d'Initiation et le Manuel de Référence, il existe donc une faille que le présent Manuel s'efforce de combler, pour permettre au programmeur de passer, sans douleur, du niveau débutant au niveau confirmé.

L'une des difficultés, pour le débutant, à réaliser des programmes d'un niveau non élémentaire, résulte de la nécessité de connaitre la quasi-totalité des instructions disponibles, avec ce que l'on pourrait appeler leur mode d'emploi.

Or souvent la lecture du Manuel de Référence ne permet pas de voir à quoi peut pratiquement servir une instruction. Le lecteur attentif peut comprendre la fonction réalisée par une instruction, mais en l'absence d'exemples pratiques d'application n'en voit pas toujours l'utilité. Il ne pensera donc pas à utiliser cette instruction pour résoudre son problème car l'on ne retient bien que ce que l'on a parfaitement compris.

Prenons par exemple l'instruction PTRIG. Qu'en dit le Manuel?

PTRIG vaut -1 si le contact du crayon optique est fermé et 0 dans le cas contraire. N'importe qui est capable de comprendre cela, mais à quoi cela peut-il bien servir?

Noua verrons plus loin une utilisation très importante de cette instruction, dans tous les programmes faisant intervenir le crayon optique. Elle permet de détecter l'usage du crayon optique pour le choix dans un "menu" de préférence à une réponse au clavier. Encore faut-il en avoir vu une application pratique pour y penser. Ce mode d'utilisation de PTRIG, une fois identifié par un exemple, est alors facilement mémorisé et le programmeur, chaque fois qu'il aura à faire intervenir le crayon optique, ira chercher l'exemple, soit dans l'un de ses précédents programmes, aoit dans notre Manuel (les programmes et les étiquettes où est utilisé PTRIG, sont donnés en annexe, dans le répertoire alphabétique dea inatructions).

Comment lire ce Manuel?

Nous n'en recommandons pas la lecture dans les transports en commun; il ne vaut pas un bon roman policier! (Le nombre de personnages est beaucoup plus grand dans le Manuel; connaissez-vous SCREEN 0,0,7, agent tellement secret qu'il ne publie ses messages qu'en noir sur fond noir et l'inquiétant SCREEN 0,7,0 qui n'écrit que des faire-parts de deuil !)

Il est bien préférable de le lire "de la main gauche", en programmant le TO7 "de la main droite". Chaque instruction doit être comprise, le Manuel y aidant en expliquant le pourquoi et le comment.

Ne vous imaginez surtout pas que l'auteur a écrit ces programmes exemples tout d'une traite et qu'une fois tapé, le programme a marché du premier coup. Nous ne connaissons aucun programmeur confirmé qui puisse se vanter d'avoir réussi une fois dans sa carrière un "sans faute" sur des programmes de plus de dix instructions!

Chaque programme a demandé de nombreux essais, suivis de retouches et de modifications (pas toutes nécessairement heureuses).

Vous constaterez qu'il est difficile d'imaginer ce que va donner une présentation d'information à l'écran (résultats de calculs, menus proposant un choix, instructions à l'opérateur...) et que bien souvent, après essais, on est amené à déplacer les symboles, changer leurs tailles, leurs couleurs, changer la couleur du fond, etc.

Encore supposons-nous que nous avons programmé exactement ce que nous avions envisagé ! Il est tellement facile d'intervertir par étourderie les valeurs de ligne et de colonne dans un LOCATE, d'intervertir la couleur des caractères et du fond dans un SCREEN, d'écrire une condition à l'envers!

Un conseil à cette occasion. Ne perdez pas votre temps à chercher à détecter les fautes d'étourderie avant essai. Elles deviennent en général évidentes une fois l'essai effectué. Par contre faites de nombreux essais au cours de la frappe de votre programme. Chaque fois que vous avez frappé un "bloc" d'instructions "essayables", lancez le programme et effectuez les corrections nécessaires.

En avançant ainsi pas à pas vous "déverminerez" progressivement votre programme, ce qui est beaucoup plus facile qu'un déverminage global, car les erreurs peuvent se masquer les unes les autres. En outre la recherche des erreurs est facilitée si vous savez qu'elles ne peuvent se situer que dans les quelques instructions non encore essayées.

Il peut être également commode de placer des instructions provisoires au cours de la mise au point, en particulier pour vérifier que les sauts conditionnels s'effectuent comme prévu.

Par exemple une instruction ON X GOTO ou une instruction ONPEN GOTO peut être vérifiée en plaçant aux étiquettes de destination des instructions provisoires premettant de vérifier que les branchements se font correctement.

Nous donnons des exemples de ces techniques de mise au point tout au long du Manuel. Aussi est-il nécessaire de suivre le Manuel tout en programmant le TO7, plutôt que de taper le programme définitif et constater (aux erreurs de frappe près) que le programme tourne.

Vous constaterez enfin que les mêmes conseils ou les mêmes remarques se retrouvent dans les différents chapitres. Ceci est volontaire. Nous pensons qu'il est nécessaire de répéter plusieurs fois la même chose pour être sûr d'être bien perçu. Par ailleurs, nous l'avons dit, ce Manuel n'est pas un roman qui se dévore en une soirée ! Une remarque importante, lue au chapitre I, peut très bien être oubliée en abordant le chapitre V, quelques jours plus tard.

La programmation n'est pas une technique difficile demandant des aptitudes intellectuelles spéciales. Par contre comme toutes les techniques, elle demande un apprentissage. On n'apprend pas à skier, à conduire une voiture, à parler une langue étrangère en cinq minutes. Il faut travailler, et seule la pratique permet d'acquérir les connaissances nécessaires.

Puisse ce Manuel vous aider à maitriser la programmation du TO7 !

Celle-ci vous reserve des joies intellectuelles qui valent largement celles du bridge, des échecs ou du scrabble.

Pour faciliter la lecture de ce Manuel, les références au Manuel de Référence du BASIC TO7 seront notées sous la forme:

-(M.Ref n)-

où n est le numéro de la page du Manuel de Référence.

Les références au présent Manuel, POUR ALLER PLUS LOIN EN BASIC TO7, seront notées sous la forme

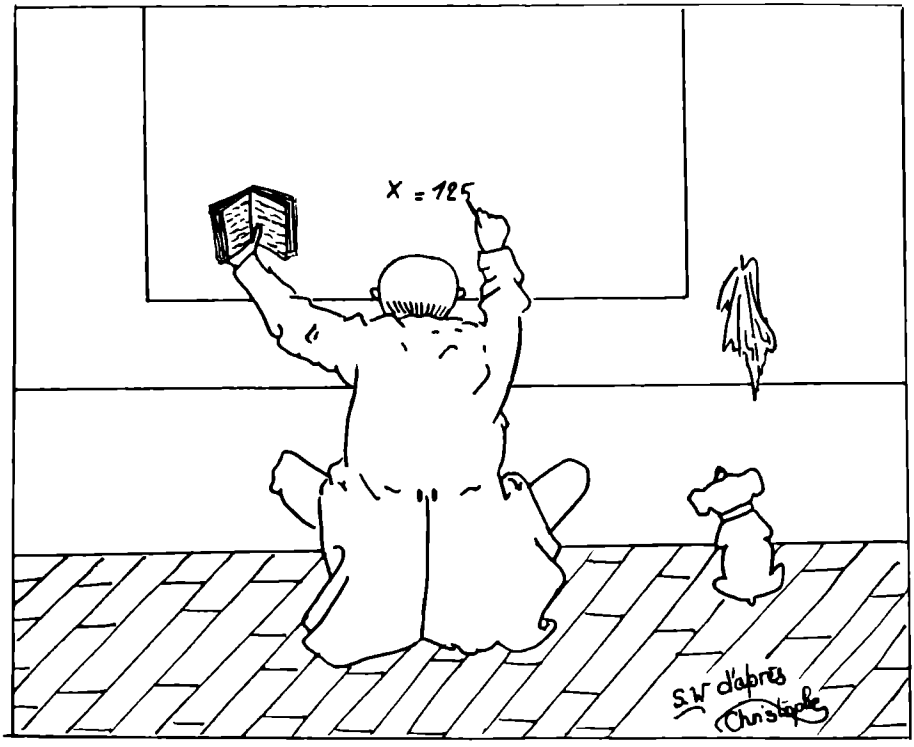
-(M. Bas n)-

Table des matières

Avertissement VII

1. Présentation à l'écran des caractères disponibles 1
2. Choix multiple. Présentation d'un menu 9
3. Test musique 33
4. Mire 45
5. Conception de caractères graphiques 63
6. Clavier 105
7. Miniécriture 127
8. Calculette 147
9. Tableau de données 169
10. Sauvegarde des données 205
11. Calendrier 229

A 1. Contrôle des couleurs 251
A 2. Programmation de l'Imprimante PR 90 080 273
A 3. Index des instructions 291



Où COSINUS préfigure, génialement, la copie d'une ROM sur une RAM.

1

Présentation à l'écran des caractères disponibles

Nous allons commencer par un programme simple destiné à présenter sur l'écran l'ensemble des caractères disponibles en BASIC TO7.

Nous savons qu'à chaque caractère correspond un numéro de code (code ASCII) compris entre 0 et 127 -(M.Ref 219)-.

Les codes compris entre 0 et 31 correspondent aux caractères "Controle" obtenus en tapant la touche CNT puis l'une des touches Lettres (de A à Z). Quelques-uns de ces caractères "Controle" sont obtenus directement en tapant les touches de commandes comme les flèches ou les touches STOP, INS, EFF, RAZ. Ces numéros de code ne donnent pas lieu à une impression de caractère sur l'écran. Nous en verrons une utilisation dans le programme CLAVIER au chapitre suivant.

Nous ne nous intéresserons donc qu'aux caractères correspondant aux numéros de code compris entre 32 et 127.

Par ailleurs quelques caractères sont bâtis en tapant la touche ACC puis une seconde touche appropriée.

Nous nous proposons donc d'afficher à l'écran les numéros de code ASCII suivis du caractère correspondant.
Pour cela nous utiliserons la fonction CHR$(N) -(M.Ref 113)- qui fournit le caractère correspondant au numéro de code ASCII N.

Enfin les caractères spéciaux obtenus en tapant au préalable la touche ACC seront obtenus en concaténant le caractère V$ = CHR$(22) (le numéro de code ASCII correspondant à la touche ACC est 22) avec le caractère CHR$(N).

Nous verrons plus loin à quelque peu modifier la construction de ces caractères dans le cas des lettres accentuées; mais il est préférable de s'en tenir pour l'instant à cette méthode et de ne la modifier que suite à l'expérience.

Nous avons à disposer sur l'écran 127 - 31 = 96 caractères.

Nous les présenterons donc en quatre colonnes de 24 lignes en affichant dans chaque colonne le numéro de code ASCII, le caractère correspondant et, s'il existe, le caractère spécial obtenu avec la touche ACC.
Sur la première ligne nous présenterons les numéros 32, 32 + 24 = 56, 56 + 24 = 80, 80 + 24 = 104 et sur la première colonne les numéros 32 à 55.

Notre écran se présentera donc de la façon suivante :

32	56	80	104
33	57	81	105
.....			
.....			
54	78	102	126
55	79	103	127

Le numéro de code ASCII N variant de 32 à 127 sera affiché en colonnes 0, 10, 20 et 30 à l'aide de l'instruction PRINT USING -(M.Ref 90)- avec une image comportant 3 places pour les 3 digits constituant le nombre N. L'image utilisée est donc constituée de 3 dièses. Ainsi le numéro 32 en première ligne s'affichera avec le digit 3 en colonne 1, le premier digit étant un espace situé en colonne 0.

Le caractère correspondant au numéro de code ASCII N sera affiché avec un décalage de 2 espaces par rapport au dernier digit de N, et le caractère spécial, s'il existe, avec un décalage de un espace par rapport au caractère.

Il nous reste, maintenant que ces principes de présentation ont été adoptés, à examiner le programme qui permet de réaliser notre objectif.

Nous proposons le programme suivant :

```
10 CLS:SCREEN 4,6,6

20 VQ=24:DX=10:TD=32:A$=CHR$(22)

30 FOR I=32 TO 55

40 FOR K=0 TO 3

50 N=I+VQ*K:KK=DX*K:LL=I-TD

60 COLOR 5:LOCATE KK,LL,0:PRINT USING "###";N:
   COLOR 4:LOCATE KK+5,LL:PRINT CHR$(N):COLOR 0:
   LOCATE KK+7,LL:PRINT A$+CHR$(N)

70 NEXT K,I

80 R$=INPUT$(1):IF ASC(R$) <> 13 GOTO 80

90 SCREENPRINT

100 COLOR 4:CLS
```

*** 10

Efface l'écran. Les caractères seront Bleu (4), le fond Cyan (6) et l'entourage Cyan (6) (on se reportera à l'annexe I pour mieux comprendre l'utilisation des instructions gérant les couleurs).

*** 20

Définit les trois constantes VQ (valeur 24), DX (valeur 10), TD (valeur 32) et la chaîne A$ (caractère controle 22, c'est à dire caractère correspondant à la touche ACC).

Ces quatre paramètres nous seront utiles dans la suite du programme.

Il est recommandé de ne pas utiliser des constantes, sous forme numérique, à l'intérieur du programme car cela impose à l'interprèteur, au moment de l'exécution de l'instruction, de transcrire la valeur, représentée dans l'instruction par une chaîne de caractères chiffres, en une valeur numérique utilisable dans les calculs.

Rappelons que l'interprèteur est le logiciel qui transforme chaque instruction, écrite en BASIC, en langage machine directement assimilable par le microprocesseur.

Si la constante est utilisée à l'intérieur d'une boucle, l'interprèteur doit effectuer la transcription à chaque passage sur l'instruction, ce qui conduit à une perte de temps inutile.

Dans notre exemple la boucle est relativement courte et le gain de temps n'est pas significatif, mais nous avons préféré utiliser cette méthode de déclaration préalable des constantes pour en rappeler le principe.

Nous noterons enfin que nous n'avons pas utilisé des variables A, B ou C pour les constantes 24, 10 et 32 mais les variables VQ, DX et TO qui ont un sens mnémonique qui évite des erreurs de programmation ultérieures.

*** 30
Déclaration de boucle portant sur le paramètre I. On fait varier I de 32 à 55 ce qui représente les numéros de code ASCII de la première colonne.

*** 40
Déclaration de boucle portant sur le paramètre K. On fait varier K de 0 à 3, K représentant le numéro de la rangée verticale (0 pour la première, 3 pour la dernière).
Ainsi le numéro de code ASCII N figurant dans la rangée K sur la ligne où est affiché, en colonne 0, le numéro I, est-il N = I + 24 * K

*** 50
a/ Calcul du numéro de code ASCII N par la formule démontrée ci-dessus.

b/ Le numéro de colonne écran correspondant à la rangée K est KK = 10 * K. C'est en colonne KK que l'on affichera le premier chiffre de N (ou un espace si N ne possède que deux chiffres)

c/ Enfin le numéro de code ASCII 32 doit être affiché sur la ligne 0, ce qui impose un numéro de ligne LL = I - 32 pour le numéro de code I.

*** 60

a/ Fixe la couleur des caractères à 5 (Magenta) (voir Annexe I)

b/ Positionne le curseur en colonne KK, en ligne LL.

Le zéro figurant en troisième paramètre de l'instruction LOCATE efface le curseur -(M.Ref 72)-

c/ Affiche à l'écran la valeur de N, en plaçant le premier caractère (qui peut être un espace si N n'a que deux chiffres) à la position du curseur définie par l'instruction LOCATE précédente.

d/ Fixe la couleur des caractères à 4 (Bleu)

e/ Positionne le curseur en colonne KK + 5, c'est à dire 2 espaces après la valeur de N, puisque N est affiché par l'instruction PRINT USING précédente avec 3 caractères.

f/ Affiche le caractère correspondant au numéro de code ASCII N.

g/ Fixe la couleur des caractères à 0 (Noir)

h/ Positionne le curseur en colonne KK + 7, c'est à dire un espace après le caractère de code N.

i/ Affiche, s'il existe, le caractère spécial bâti avec la touche ACC préalablement tapée.

En conséquence on affiche en Magenta le numéro de code, en Bleu le caractère correspondant, en Noir le caractère spécial.

*** 70

Ferme les deux boucles en I et K. Cette instruction est équivalente aux deux instructions successives NEXT I:NEXT K.

* * * * * *

Avant d'examiner la suite du programme, il est faut essayer cette première partie, qui constitue un "bloc" exécutable puisque la dernière instruction a cloturé les deux boucles.

Nous constatons après essai que le programme marche presque correctement !

Néanmoins on relève cinq anomalies, apparemment mystérieuses. Les numéros de code 89, 90, 91 et 96 sont décalés d'un espace vers la gauche et un ç s'est accolé au numéro 99.

Nous sommes là en présence d'un "bug" qui peut laisser le programmeur débutant quelque peu désemparé ! (le terme anglo-saxon "bug" signifie vermine).

Bien entendu l'auteur lui-même est tombé dans le piège et il a dû rechercher l'origine des anomalies.

On constate que les anomalies se produisent sur les lignes où ont été affichées les caractères spéciaux correspondant aux trois accents, au tréma et au caractère ç. Or un accent ou un tréma n'existe pas "seul", il s'applique à un autre caractère. Il est donc nécessaire de faire figurer derrière le groupe CHR$(22) + CHR$(N) un troisième caractère. Il se produit un phénomène analogue avec la cédille. Aussi allons nous modifier la dernière instruction de 60 sous la forme PRINT V$ + CHR$(N) + " ", c'est à dire en rajoutant un espace après le caractère spécial. Dans ces conditions l'effet de l'accent qui ne provoque pas le déplacement du curseur est annulé par la présence de l'espace.

* * * * * *

*** BD

a/ Place dans la variable R$ le premier caractère tapé au clavier -(M.Ref 174)-.

Il faut noter que l'instruction INPUT$(1) permet de charger dans la variable à gauche du signe = n'importe quel caractère tapé au clavier, en particulier le caractère correspondant à la touche ENTREE (numéro de code ASCII égal à 13). Par contre la fonction INPUT -(M.Ref 65)- ne permet pas de capter tous les caractères.

b/ Teste le caractère contenu dans R$.

La fonction ASC(R$) -(M.Ref 111)- retourne le numéro de code ASCII du premier caractère de R$ (dans notre cas R$ n'a qu'un seul caractère).

Si ce numéro est différent de 13 (code correspondant à la touche ENTREE) le programme retourne à l'instruction 80 pour attendre une nouvelle frappe au clavier. En conséquence le programme ne passe à l'instruction suivante 90 que si l'opérateur tape la touche ENTREE, les autres touches étant inopérantes.

Cette instruction, que nous retrouverons de nombreuses fois, permet de stopper le programme pour permettre à l'opérateur de consulter les résultats affichés à l'écran.

La poursuite du programme s'effectue en tapant une touche précise, ici la touche ENTREE.

On pourrait se dispenser du test, auquel cas la frappe d'une touche quelconque permettrait la poursuite du programme. Il est toutefois prudent, dans de nombreux cas, d'imposer la frappe d'une touche précise pour éviter des poursuites intempestives du programme sur fausse manoeuvre.

*** 90

Cette instruction n'est utilisable que si l'ordinateur est connecté avec l'imprimante thermique PR 90-040.

Elle provoque la recopie de l'écran, ce qui permet de disposer d'un document rappelant en particulier comment obtenir les caractères spéciaux.

Notons que l'imprimante à aiguilles PR 90-080 n'accepte pas l'instruction SCREENPRINT.

*** 100

Fixe la couleur des caractères à 4 (Bleu).

Cette instruction est utile pour retrouver l'écran habituel (caractères Bleu, fond et entourage Cyan) après effacement par l'instruction suivante CLS

Le programme définitif est donc le suivant :

```
10 CLS:SCREEN 4,6,6

20 VQ=24:DX=10:TD=32:A$=CHR$(22)

30 FOR I=32 TO 55

40 FOR K=0 TO 3

50 N=I+VQ*K:KK=DX*K:LL=I-TD

60 COLOR 5:LOCATE KK,LL,0:PRINT USING "###";N:
   COLOR 4:LOCATE KK+5,LL:PRINT CHR$(N):COLOR 0:
   LOCATE KK+7,LL:PRINT A$+CHR$(N)+" "

70 NEXT K,I

80 R$=INPUT$(1):IF ASC(R$) <> 13 GOTO 80

90 SCREENPRINT

100 COLOR 4:CLS
```

2

Choix multiple. Présentation d'un menu

Il arrive bien souvent qu'au cours du déroulement d'un programme, on soit amené à proposer un choix à l'opérateur.

Il est commode de proposer les diverses options possibles sous forme d'une liste que l'on appelle habituellement un "menu".

Nous nous proposons d'établir un programme simple, montrant les diverses techniques de programmation permettant de faciliter les manipulations de l'opérateur et de réduire les erreurs de choix.

Le thème retenu sera le suivant : on propose à l'opérateur une liste de cinq chansons. Une fois choisie un titre le TO7 joue les premières mesures de la chanson.

Ce thème n'a bien entendu d'intéret que dans la mesure où il nous permet de présenter les techniques de "menu", quelques techniques de présentation d'information à l'écran et la methode pour faire "chanter" notre TO7 !

Le programme peut être divisé en plusieurs blocs qui seront mis au point successivement :

- un bloc "présentation du menu" à l'écran. Un souci de clarté et même d'esthétique doit prévaloir.

- un bloc "logique" analysant la réponse de l'opérateur pour orienter le programme sur la chanson choisie.

- un bloc "musique" permettant de jouer l'air retenu.

Il ne faudra pas enfin oublier de proposer à l'opérateur un dernier choix, une fois la musique exécutée, à savoir s'il veut choisir une nouvelle chanson ou s'il veut clôturer (les informaticiens utilisent bien souvent le terme "clôturer" pour "terminer" bien que l'Académie ne lui donne pas exactement ce sens; pour avoir l'air d'informaticiens dans le vent nous utiliserons donc abondamment ce vocable)

Etudions tout d'abord le premier bloc, le bloc de présentation du menu.

Nous proposerons cinq titres de chansons, à savoir "Au clair de la Lune", "J'ai du bon tabac", "A la claire fontaine", "Sur l'pont du Nord" et "Nous n'irons plus au bois". Rien n'empêche le lecteur de modifier notre programme pour y introduire des titres plus alléchants de Brel ou de Brassens !

Nous afficherons ces cinq titres sur les lignes 2, 7, 12, 17 et 22. Nous dessinerons en bout de chaque ligne un carré contenant un numéro allant de 1 à 5.

Le choix sera effectué par l'opérateur soit en tapant au clavier le numéro de la chanson retenue, soit en désignant le carré correspondant à l'aide du crayon optique.

Nous rajouterons, à gauche et à mi-hauteur de l'écran, un carré supplémentaire contenant la lettre C pour permettre à l'opérateur de "Clôturer"

Par souci d'esthétique nous inscrirons les titres dans des rectangles blancs, bordés en dessous et à droite de lignes de couleur donnant une impression de relief. Les carrés seront de même bordés de lignes de couleur les faisant apparaître comme des touches en relief. Nous utiliserons cinq couleurs, rouge, vert, jaune, bleu, magenta pour les cinq titres et les "touches" correspondantes et le noir pour la "touche" C

Le programme correspondant à ce premier bloc peut alors s'écrire comme suit :

```
10 DATA Au clair de la Lune,J'ai du bon tabac,
   A la claire fontaine,Sur l'pont du Nord,
   Nous n'irons plus au bois:DIM Z$(4)

20 CLS:SCREEN 4,6,6

30 FOR B=0 TO 4

40  Y1=40*B+8:Y2=Y1+23:BOXF(288,Y1)-(311,Y2),-8:
    COLOR 0,7:LOCATE 37,5*B+2,0:PRINT CHR$(49+B)

50   FOR J=1 TO 3

60    YA=Y1+J:YB=Y2+J:XA=311+J:XB=288+J

70    LINE (XA,YA)-(XA,YB),B+1:LINE -(XB,YB),B+1

80   NEXT J

90 NEXT B

100 BOXF (0,88)-(23,112),-8:FOR J=1 TO 3:
    LINE (23+J,88+J)-(23+J,112+J),0:
    LINE -(J,112+J),0:NEXT:COLOR 0,7:
    LOCATE 1,12:PRINT "C"

110 FOR B=0 TO 4

120  Y1=40*B+15:Y2=Y1+8:X2=63+8*LEN(Z$):
     READ Z$:Z$(B)=Z$

130  BOXF (48,Y1)-(X2,Y2),-8

140   FOR J=1 TO 3

150    YA=Y1+J:YB=Y2+J:XA=X2+J:XB=47+J

160    LINE (XA,YA)-(XA,YB),B+1:LINE -(XB,YB),B+1

170    COLOR 0,7:LOCATE 7,5*B+2:PRINT Z$

180   NEXT J

190 NEXT B
```

*** 10

a/ Ligne de données permettant de charger les titres des cinq chansons.

Nous noterons que les données figurant dans une instruction DATA -(M.Ref 38)- sont stockées sous forme de chaînes de caractères, même si ce sont des valeurs numériques. C'est à la lecture par l'instruction READ que la donnée lue est identifiée comme étant une variable numérique ou comme une chaîne de caractères (READ Z interprète la variable lue comme une variable numérique simple précision, READ Z# comme une variable double précision, READ Z$ comme une chaîne de caractères).

On notera donc que les données sont écrites sans être placées entre guillemets bien que ce soient des chaînes de caractères.

b/ Réserve 5 places mémoires (indice variant de 0 à 4) pour la variable Z$(I) destinée à stocker les titres des chansons.

Cette réservation n'est pas obligatoire puisque nous n'utilisons que 5 valeurs (dès qu'une variable dimensionnée apparaît dans une instruction, une réservation automatique de 11 places mémoires (indice variant de 0 à 10) est effectuée si la réservation par DIM -(M.Ref 46)- n'a pas été faite dans une instruction précédente. Il est toutefois recommandé de faire cette réservation systématiquement pour économiser des places mémoires.

Il faut prendre garde enfin à placer l'instruction DIM en tout début de programme. Nous aurons en effet à revenir, une fois un air exécuté, à l'instruction de début du bloc d'affichage du "menu", c'est-à-dire à l'instruction 20. Or il est interdit de dimensionner plusieurs fois de suite la même variable par l'instruction DIM. Il ne faut donc pas placer DIM Z$(4) sous une étiquette sur laquelle le programme est appelé à passer plusieurs fois (une telle erreur est signalée par le message DD)

*** 20

a/ Efface l'écran.

b/ Les caractères seront Bleus (4), le fond Cyan (6) et l'entourage Cyan (6) -(M.Ref 193)-

*** 30

Ouverture d'une boucle sur le paramètre B. B varie de 0 à 4 (le numéro N de la chanson est égal à B + 1)

*** 4D

Les carrés auront 24 points de côté (soit 3 caractères : 3 * 8 = 24), ce qui permettra de placer un caractère au centre (chiffre de 1 à 5). Chaque chiffre doit se trouver sur la même ligne que le titre correspondant de la chanson, donc en lignes 2, 7, 12, 17 et 22. Nous les placerons en outre en colonne 37.

Ceci nous impose pour la chanson N (N = B-1) de placer le chiffre N en colonne 37, ligne 5*B+2. Par suite la "case caractère" en haut et à gauche du chiffre N doit être située en colonne 36, ligne 5*B+1. Les coordonnées du point en haut et à gauche de cette "case caractère", donc en haut et à gauche de notre carré entourant le chiffre N, sont donc X1 = 36 x 8 = 288 et Y1 = 8*(5*B+1). Les coordonnées du coin bas-droit du carré seront par suite X2 = X1 + 23 et Y2 = Y1 + 23.

Nous devons prendre garde à ce que l'abscisse X1 du coin haut-gauche de notre carré soit un multiple exact de 8 et que l'abscisse X2 du coin bas-droit soit un multiple de 8 moins 1. Nous allons en effet tracer un carré en "couleur de fond" Blanc à l'aide de l'instruction BDXF. Il est nécessaire, traçant une figure en "couleur fond" Blanc sur le fond général Cyan de respecter cette condition sous peine de voir le fond Blanc "baver" sur le fond Cyan, ce qui conduirait à un "carré" Blanc de forme rectangulaire ! On se reportera à l'Annexe I pour plus de détails sur les problèmes de gestion des couleurs.

Détaillons maintenant les diverses instructions placées sous l'étiquette 4D

a/ Calcul de l'ordonnée Y1 du coin haut-gauche du carré

b/ Calcul de l'ordonnée Y2 du coin bas-droit du carré

c/ Dessin d'une "boîte pleine" de couleur -8 (Couleur de fond Blanc) -(M.Ref 187)-

d/ Fixe la couleur des caractères au Noir (0) avec une couleur de fond Blanc (7). Les caractères affichés par les PRINT suivants seront donc Noirs sur fond Blanc.

e/ Positionne le curseur en colonne 37 et en ligne 5*B+2.

Le zéro comme troisième paramètre du LOCATE supprime le curseur.

f/ Affiche le caractère ayant le numéro de code ASCII 49+B (voir fonction CHR$ -(M.Ref 113)-).

Ainsi le caractère "1" (numéro de code ASCII 49) est affiché pour B = 0; pour B = 4 c'est le caractère "5" (numéro de code ASCII 53) qui est affiché.

*** 50
Ouverture d'une boucle en J.

J est un paramètre variant de 1 à 3, destiné à compter le nombre de lignes tracées au-dessous et à droite du carré pour donner l'impression de relief.

*** 60
Nous traçons quatre lignes à droite du carré, comprises entre les ordonnées YA = Y1 + J (Y1 est l'ordonnée du haut du carré) et YB = Y2 + J (Y2 est l'ordonnée du bas du carré) et d'abscisses XA = 311 + J (311 est l'abscisse du bord droit du carré).

Cela donne quatre lignes en dégradé vers la droite et le bas.

On complète le dessin par quatre lignes horizontales en dégradé analogue : coordonnées d'origine identiques aux coordonnées d'extrémité des lignes verticales précédentes, coordonnées d'extrémité XB = 288 + J (288 est l'abscisse du bord gauche du carré), YB = Y2 + J (Y2 est l'ordonnée du bas du carré).

a/ Calcul de l'ordonnée YA

b/ Calcul de l'ordonnée YB

c/ Calcul de l'abscisse XA

d/ Calcul de l'abscisse XB

*** 70

a/ Tracé d'une ligne verticale entre les points XA,YA et XA,YB en couleur égale à B+1, c'est-à-dire en couleur de graphisme (voir Annexe I), puisque B+1 est positif (couleur Rouge (1) pour le premier carré, couleur Magenta (5) pour le cinquième).

b/ Tracé d'une ligne horizontale entre le point XA,YB (retenu par défaut puisque l'instruction après le mot LINE ne comporte pas de première parenthèse) et le point XB,YB. La couleur est la même, B+1, que pour la ligne verticale précedente.

*** 80 - 90

Fermeture des boucles en J et en B. On aurait pu simplifier et écrire une seule instruction sous la forme : 90 NEXT J,B mais la programmation aurait perdu de la clarté à la relecture.

*** 100

Ce groupe d'instructions sous la même étiquette, a pour but de dessiner le carré situé à gauche et à mi-hauteur de l'écran, contenant le caractère "C". Le caractère "C" est situé en colonne 1, ligne 12. Les instructions sont analogues à celles décrites de 30 à 90.

*** 110

Ouverture d'une boucle en B. B est encore le paramètre de comptage des lignes d'inscription des titres de chansons.

*** 120 - 180

Nous allons tracer des rectangles en couleur fond Blanc sur les lignes 2, 7, 12, 17, 22 sur lesquels nous écrirons les titres des chansons. Chaque rectangle sera mis en relief par quatre lignes verticales à droite, en dégradé, et quatre lignes horizontales au-dessous, en degradé également.

La programmation est analogue à celle utilisée pour les carrés.

La seule différence réside dans la lecture des données DATA par l'instruction READ en 120, qui place dans la variable Z$ les chaînes successives représentant les différents titres et le calcul de la "longueur" du rectangle blanc à partir de la longueur de la chaîne Z$ par la fonction LEN(Z$) -(M.Ref 128)-.

Enfin la chaîne Z$ est mise en mémoire dans la variable Z$(B) ce qui nous sera utile pour afficher le titre de l'air choisi pendant son exécution.

*** 190
Clôture de la boucle en B

Il est temps, une fois ce programme tapé, de l'essayer.

Il constitue en effet un bloc exécutable et, comme nous l'avons dit dans l'Avertissement aux Lecteurs, il est très important de faire "tourner" dès que possible les parties de programme exécutables pour y détecter les erreurs, erreurs de frappe donnant des SN Errors (Erreurs de Syntaxe) et erreurs de logique.

Le programme qui vous est proposé ici est exempt de ce dernier type d'erreur mais il ne faut pas croire qu'il a été conçu directement sans erreur ! Il est si facile de se tromper dans un LOCATE ou le calcul des coordonnées d'un point. Ce n'est qu'en observant le résultat sur l'écran que l'on détecte de façon évidente ce type d'erreur.

Passons maintenant au deuxième bloc, le bloc logique analysant la réponse de l'opérateur pour orienter le programme vers le bloc d'exécution de la musique choisie.

Pour bien le différencier du bloc précédent, nous le commencerons à l'étiquette 500. Cette remarque peut paraître futile. Elle est en réalité très importante, car une telle pratique facilite grandement la relecture et la mise au point du programme.

Nous nous proposons de capter la réponse de l'opérateur sous l'une ou l'autre des deux formes suivantes :

a/ Frappe au clavier d'un chiffre, compris entre 1 et 5, ou de la lettre C

b/ Pointage à l'aide du crayon optique de l'une des 6 "touches" dessinées sur l'écran (les 5 touches carrées marquées 1 à 5 et la touche C).

La captation d'un caractère "à la volée" se fait par l'instruction INKEY$ -(M.Ref 123)- . Si une touche a été frappée la variable R$ définie par l'instruction :

R$ = INKEY$

contient le caractère correspondant à cette touche. Si au préalable la variable R$ a été "mise à zéro" (chaîne de caractères de longueur nulle), un test sur le contenu, non nul ou nul, de R$ nous renseignera sur le fait que l'opérateur a répondu ou non par l'intermédiaire du clavier.

Le fait que l'opérateur ait répondu par l'intermédiaire du crayon optique est détecté grâce à l'instruction PTRIG.

Si le crayon optique a été appuyé sur l'écran (ou sur toute autre surface, nous verrons comment faire la distinction plus loin), le paramètre PTRIG prend la valeur -1 (Variable logique VRAI), sinon il prend la valeur 0 (Variable logique FAUX). Un test sur PTRIG nous permettra donc de détecter une réponse par le crayon optique.

Il nous reste ensuite à tester si la réponse fait partie des réponses attendues (nombre compris entre 1 et 5 ou caractère C pour la réponse au clavier, zone sur l'écran correspondant à l'une des "touches" dessinées)

Pour repérer sur quelle "touche" le crayon optique a été placé, il nous faut identifier les zones correspondant aux touches par l'instruction PEN -(M.Ref 84)- .

Une instruction PEN définit une zone rectangulaire avec les coordonnées des coins haut-gauche et bas-droit du rectangle. Sa structure est analogue à celle de l'instruction BOX.

Il est commode d'insérer ces instructions dans le programme de dessin des touches au moment où on les définit par l'instruction BOX. Nous modifierons donc les instructions 40 et 100 en les complétant par une instruction PEN.

```
40  Y1=40*B+8:Y2=Y1+23:BOXF(288,Y1)-(311,Y2),-8:
    COLOR 0,7:LOCATE 37,5*B+2,0:PRINT CHR$(49+B):
    PEN B:(288,Y1)-(311,Y2)
```

```
100 BOXF (0,88)-(23,112),-8:FOR J=1 TO 3:
    LINE (23+J,88+J)-(23+J,112+J),0:
    LINE -(J,112+J),0:NEXT:COLOR 0,7:
    LOCATE 1,12:PRINT "C":PEN 5:(0,88)-(23,112)
```

*** 40

L'instruction PEN, que l'on vient de rajouter,définit donc le carré dessiné par l'instruction BOXF précédente (les coordonnées sont les mêmes pour les deux instructions) avec le numéro repère 8, c'est-à-dire D pour la "touche" du haut et 4 pour la "touche" du bas.

On notera qu'il est important de commencer la numérotation des "zones" par la valeur D, car la première étiquette adressée par l'instruction ONPEN se réfère à la "zone" 0 (il y a donc une différence notable avec l'instruction ON X GOTO où la première étiquette adressée se réfère à la valeur 1 du paramètre X).

*** 100

L'instruction supplémentaire attribue le numéro repère 5 à la "touche" C.

Il faut enfin sur une réponse plausible brancher le programme sur le bloc "musique" pour exécuter l'air demandé. Des instructions ONPEN -(M.Ref B2)- et ON X -(M.Ref BD)- nous permettront ces branchements.

Le programme répondant à ces diverses exigences est alors le suivant.

```
500 R$="":R$=INKEY$

510 IF PTRIG GOTO 600

520 IF R$="" GOTO 500

530 R=ABC(R$):
    Z=1-(R)48 AND R(54)*(R-48)-6*(R$="C")

540 ON Z GOTO 500,1000,2000,3000,4000,5000,6000

600 ONPEN GOTO 1000,2000,3000,4000,5000,6000

610 GOTO 500
```

*** 500

a/ Met à "zéro" la variable R$

b/ Recueille dans R$ le ceractère éventuellement frappé eu clevier

*** 510

PTRIG étant une variable logique est testée directement per la condition IF. Si PTRIG est VRAI (le crayon optique a été appuyé), alors le programme est branché en 600. Dans le cas contraire (PTRIG = FAUX) on passe à l'instruction suivante. Ainsi l'utilisation du crayon optique est détectée dès que l'interrupteur du creyon est fermé par appui sur l'écran et le programme est brenché sur le reconnaissance en 600 de la zone d'écran vue par le crayon.

*** 520

Le crayon n'ayant pes été utilisé, c'est la variable R$ qui est testée. Si R$ est égale à "zéro" (chaîne nulle = "") elors le programme est brenché en 200 en attente d'une frappe de touche ou d'une utilisetion du creyon. Il faut noter que contrairement eux instructions INPUT et INPUT$(N) le programme ne s'arrête pes sur une instruction INKEY$. Il capte "à la volée" le cerectère tapé au clavier. En conséquence le progremme boucle en permanence entre 500 et 520 jusqu'à ce qu'une touche soit freppée ou le creyon optique appuyé.

*** 530

e/ La varieble R contient le numéro de code ASCII de R$, ce qui va permettre d'identifier la touche tapée.

b/ Z est une variable calculée à l'aide de variables logiques testant R ou R$.

On sait que la veriable (X = A) prend la valeur -1 si la condition entre perenthèses est vérifiée (si X est égal à A), et prend la valeur 0 si la condition n'est pas vérifiée (si X est différent de A) -(M.Ref 20)-.

En conséquence si R$ est un caractère correspondant aux chiffres de 1 à 5 alors R (numéro de code ASCII de R$) est compris entre 48 et 54, bornes non comprises (le numéro de code ASCII de "1" est 49 et celui de "5" est 53). Per suite si R$ est un caractère différent des cinq caractères "1" à "5" la variable logique (R)48 AND R(54) est égale à zéro et elle est égale à -1 dans le cas contraire.

De même la variable logique (R$="C") est égale à -1 si l'opérateur a tapé la touche "C" et égale à zéro pour toute autre touche.

Ainsi la variable Z prend les valeurs 2, 3, 4, 5 ou 6 si R$ est égal à "1", "2", "3", "4" ou "5".Notons que R-48, qui n'est pas une variable logique prend les valeurs 1 à 5 lorsque R varie de 49 (numéro de code ASCII de "1") à 53 (numéro de code ASCII de "5").

Enfin Z prend la valeur 7 si R$ est un "C"et la valeur 1 dans tous les autres ces.

Cette variable Z va donc nous permettre d'une pert de rejeter toute réponse erronnée (valeur 1) et d'identifier les bonnes réponses par une valeur variant de 2 à 7.

*** 540

Suivant les valeurs de Z l'instruction ON Z GOTO -(M.Ref 80)- branche sur l'étiquette 500 (Z = 1 , cas de la réponse érronnée) pour recueil d'une nouvelle réponse, sur les étiquettes 1000 à 5000 (Z variant de 2 à 6, touches "1" à "5") ou sur l'étiquette 6000 (Z = 7, touche "C")

Nous placerons donc les instructions de "clôture" du programme à partir de 6000 et les instructions de musique en 1000, 2000, ..., 5000 (1000 pour jouer "Au clair de la Lune",..., 5000 pour "Nous n'irons plus au bois").

*** 600

L'instruction 600 n'est atteinte que si la condition PTRIG = VRAI a été vérifiée en 510. Dans ce cas nous savons que le crayon optique a été utilisé. Il nous reste maintenant à identifier la zone désignée.

L'instruction ONPEN GOTO -(M.Ref 82)- branche sur la première étiquette (1000), si la zone repérée 0 a été désignée, sur la seconde étiquette (2000), si la zone repérée 1 a été désignée, etc (les repères de zone ont été définis par les instructions modifiées 40 et 100).

Per conséquent si le cerré, numéroté 1 (carré supérieur correspondant à "Au clair de la Lune"), et repéré per le nombre 0 a été désigné, le programme est brenché sur l'étiquette 1000.

Si par contre le creyon optique a été appuyé en dehors des zones repérées (de 0 à 5 dens notre cas), alors le programme passe à l'instruction suivante, ici 610

*** 610
Aucune zone repérée de 0 à 5 n'ayant été désignée, le programme est rebranché en 500 pour lecture d'une nouvelle réponse de l'opérateur.

Il nous reste meintenant à tester le bloc logique, pour vérifier qu'il n'y a pas d'erreurs, en particulier d'erreurs d'adressage. Il nous faut donc tester que, par exemple, la frappe de la touche "2" envoie bien le programme en 2000 ou que la frappe de la touche "C" envoie bien en 6000.

Si nous essayons le programme tel quel nous aurons quelques déboires car les instructions ONPEN GOTO et ON Z GOTO ne pourront s'exécuter puisque les étiquettes 1000, 2000, ..., 6000 n'existent pas. Nous obtiendrons un message d'erreur UL (Undefined Line, Numéro de Ligne non définie).

Pour tester notre programme nous créerons des instructions provisoires 1000, 2000, ..., 6000. Par exemple nous pouvons utiliser les instructions suivantes :

```
1000 CLS:LOCATE 5,12:PRINT "1000":GOTO 10000

2000 CLS:LOCATE 5,12:PRINT "2000":GOTO 10000

3000 CLS:LOCATE 5,12:PRINT "3000":GOTO 10000

4000 CLS:LOCATE 5,12:PRINT "4000":GOTO 10000

5000 CLS:LOCATE 5,12:PRINT "5000":GOTO 10000

6000 CLS:LOCATE 5,12:PRINT "6000":GOTO 10000

10000 R$=INPUT$(1):RESTORE:GOTO 20
```

Si par exemple on frappe la touche "2" ou si l'on désigne le carré "2" à l'aide du crayon optique, on devra alors obtenir l'effacement de l'écran (CLS) et l'affichage du message "2000" en colonne 5, ligne 12 (LOCATE 5,12), ce qui confirmera un branchement correct en 2000.

Le branchement en 10000, une fois l'affichage de vérification effectué, permet de stopper le programme par l'instruction INPUT$(1). La frappe d'une touche quelconque relance le programme en branchant en 20 pour rétablir le "menu".

Il est toutefois nécessaire de prendre la précaution d'insérer l'instruction RESTORE -(M.Ref 100)- sous peine d'obtenir une erreur 00 (Out of Data) en 120. Il est indispensable en effet de remettre le pointeur de DATA à zéro avant de relire les données par READ Z$.

Dans le cas de notre programme très simple ce type de vérification du bloc logique est peut être superfétatoire, mais il peut se révéler indispensable dans des programmes plus lourds pour lesquels des erreurs de branchements peuvent provoquer des phénomènes bizarres difficiles à analyser.

Il nous reste maintenant à bâtir le bloc "musique".

Il nous faut pour cela définir la chaîne de caractères représentant la succession des notes de chacun des airs. Chaque chaîne sera bâtie sous l'étiquette correspondant à chaque titre (1000 à 5000) puis le programme sera branché à un blos commençant en 10000, comportant l'instruction PLAY et le retour au "menu" en 20.

Le bloc "musique" se présente donc ainsi :

```
1000 M$ = "O4L48T3;DODODORE;L72;MI;L12;P;L72;RE;
          L24;P;L48;DOMIREREDO"

1010 Z$ = Z$(0)

1020 GOTO 10000

2000 M$ = "O4L48T3;DOREMIDORE;P;REMIFA;P;FA;P;MI;
          P;MI;P;DOREMIDORE;P;REMIFA;P;SO;P;DO"

2010 Z$ = Z$(1)

2020 GOTO 10000

3000 M$ = "O4L48T4;DO;L24;DOMIMIRE;L48;MI;L72;DO;
          L12;P;L48;DO;L24;DOMIMIRE;L72;MI;L12;P;
          L48;MI;L24;MIREDOMISO;L72;MI;L12;P;
          L48;SO;L24;SOMIDOMI;L72;RE;L12;P;L48;
          DO;L24;DOMIMI;L12;REDO;L24;MI;L48;DO;
          L12;P;L48;MI;L24;MI;L12;REDO;L24;
          MIRE;L48;DO"
```

```
3010 Z$ = Z$(2)

3020 GOTO 10000

4000 M$ = "O3T4L24;SO;O4;DORE;L48;MIMI;L24;
          MIREMIFA;L48;MI;L24;DOMIDO;O3L48;SO;
          O4;RE;L24;REDOREMIDO"

4010 Z$ = Z$(3)

4020 GOTO 10000

5000 M$ = "O3T4L24;SOSI;O4;DO;L48;RESORE;L24;P;
          MIREDO;O3L48;SILASO"
         :M$ = M$ + "P" + M$

5010 Z$ = Z$(4)

5020 GOTO 10000

6000 CLS:SCREEN 4,6,6:END

10000 CLS:LOCATE 20-LEN(Z$)/2,12:PRINT Z$

10010 PLAY M$

10020 RESTORE:GOTO 20
```

*** 1000
La chaîne M$ contient les notes de "Au clair de la Lune".

Nous avons séparé les indications de durée de note (L), d'octave (O), de mesure (T) et les pauses (P) par des points-virgules (;) ceci à seule fin de clarifier la lecture de la chaîne bien que cela ne soit pas indispensable.

On trouvera, dans le programme exposé au chapitre suivant, plus de détails sur la composition des chaînes de caractères représentant un air de musique.

*** 1010
La variable Z$ contient le titre Z$(O) de l'air choisi.

*** 1020

Branchement en 10000 pour l'exécution de la musique

*** 2000 - 5020
Instructions analogues pour les quatre autres chansons.

On notera toutefois qu'en 5000 on a simplifié l'écriture de la chaîne M$ par une concaténation permettant de répéter deux fois la même séquence avec séparation par une pause "p".

*** 6000
Instructions de clôture.

a/ Efface l'écran

b/ L'instruction SCREEN redonne à l'écran sa configuration initiale, caractères en Bleu (4), fond et entourage en Cyan (6).

Cette instruction est indispensable car la couleur de fond a été fixée au Blanc (7) et celle des caractères au Noir (0) en 40. Son absence conduirait à attequer le programme suivant evec des caractères Noirs sur un fond Blanc.

c/ L'instruction END est indispensable pour éviter que le programme ne continue en séquence et passe à l'instruction 10000 !

*** 10000
a/ Efface l'écran donc supprime le "menu"

b/ Pour centrer au milieu de la ligne le titre de l'air choisi, il est nécessaire de calculer la position du premier caractère de la chaîne Z$.

Le formule est évidente si l'on se souvient que l'écran dispose de 40 colonnes numérotées de 0 à 39 (la fonction LEN(Z$) -(M.Ref 128)- fournit le nombre de caractères de la cheîne Z$). Le titre sera donc centré sur la ligne 12.

c/ Affiche Z$.

*** 10010
L'instruction PLAY -(M.Ref 201)- permet de "jouer" l'air stocké dans le chaîne M$.

*** 10020

a/ Comme nous l'avons déja précisé plus haut, à l'occasion du programme provisoire de test du bloc logique, il est indispensable de remettre le pointeur de DATA à zéro avant de relire les données en 120.

Le retour au "menu" doit donc être précédé d'une instruction RESTORE -(M.Ref 100)-.

b/ Retour à l'affichage du "menu" en 20. On notera, comme nous l'avons fait remarquer plus haut que le branchement se fait après l'étiquette 10 contenant l'instruction DIM pour éviter l'erreur de redimensionnement de Z$(I).

Le programme complet est donc le suivant :

```
10 DATA Au clair de la Lune,J'ai du bon tabac,
   A la claire fontaine,Sur l'pont du Nord,
   Nous n'irons plus au bois:DIM Z$(4)

20 CLS:SCREEN 4,6,6

30 FOR B=0 TO 4

40  Y1=40*B+8:Y2=Y1+23:BOXF(288,Y1)-(311,Y2),-8:
    COLOR 0,7:LOCATE 37,5*B+2,0:PRINT CHR$(49+B):
    PEN B;(288,Y1)-(311,Y2)

50   FOR J=1 TO 3

60    YA=Y1+J:YB=Y2+J:XA=311+J:XB=288+J

70    LINE (XA,YA)-(XA,YB),B+1:LINE -(XB,YB),B+1

80   NEXT J

90 NEXT B

100 BOXF (0,88)-(23,112),-8:FOR J=1 TO 3:
    LINE (23+J,88+J)-(23+J,112+J),0:
    LINE -(J,112+J),0:NEXT:COLOR 0,7:
    LOCATE 1,12:PRINT "C":PEN 5;(0,88)-(23,112)
```

```
110 FOR B=0 TO 4

120  Y1=40*B+15:Y2=Y1+8:X2=63+8*LEN(Z$):
     READ Z$:Z$(B)=Z$

130  BOXF (48,Y1)-(X2,Y2),-8

140   FOR J=1 TO 3

150    YA=Y1+J:YB=Y2+J:XA=X2+J:XB=47+J

160    LINE (XA,YA)-(XA,YB),B+1:LINE -(XB,YB),B+1

170    COLOR 0,7:LOCATE 7,5*B+2:PRINT Z$

180   NEXT J

190 NEXT B

500 R$="":R$=INKEY$

510 IF PTRIG GOTO 600

520 IF R$="" GOTO 500

530 R=ASC(R$):
    Z=1-(R)48 AND R(54)*(R-48)-6*(R$="C")

540 ON Z GOTO 500,1000,2000,3000,4000,5000,6000

600 ONPEN GOTO 1000,2000,3000,4000,5000,6000

610 GOTO 500

1000 M$ = "O4L48T3;DODODORE;L72;MI;L12;P;L72;RE;
          L24;P;L48;DOMIREREDO"
1010 Z$ = Z$(0)

1020 GOTO 10000

2000 M$ = "O4L48T3;DOREMIDORE;P;REMIFA;P;FA;P;MI;
          P;MI;P;DOREMIDORE;P;REMIFA;P;SO;P;DO"
2010 Z$ = Z$(1)

2020 GOTO 10000
```

```
3000 M$ = "O4L48T4;DO;L24;DOMIMIRE;L48;MI;L72;DO;
          L12;P;L48;DO;L24;DOMIMIRE;L72;MI;L12;P;
          L48;MI;L24;MIREDOMISO;L72;MI;L12;P;
          L48;SO;L24;SOMIDOMI;L72;RE;L12;P;L48;
          DO;L24;DOMIMI;L12;REDO;L24;MI;L48;DO;
          L12;P;L48;MI;L24;MI;L12;REDO;L24;
          MIRE;L48;DO"

3010 Z$ = Z$(2)

3020 GOTO 10000

4000 M$ = "O3T4L24;SO;O4;DORE;L48;MIMI;L24;
          MIREMIFA;L48;MI;L24;DOMIDO;O3L48;SO;
          O4;RE;L24;REDOREMIDO"

4010 Z$ = Z$(3)

4020 GOTO 10000

5000 M$ = "O3T4L24;SOSI;O4;DO;L48;RESORE;L24;P;
          MIREDO;O3L48;SILASO"
          :M$ = M$ + "P" + M$
5010 Z$ = Z$(4)

5020 GOTO 10000

6000 CLS:SCREEN 4,6,6:END

10000 CLS:LOCATE 20-LEN(Z$)/2,12:PRINT Z$

10010 PLAY M$

10020 RESTORE:GOTO 20
```

Nous avons vu deux méthodes de choix dans un "menu".

La méthode du crayon optique est de loin la meilleure, car elle ne comporte pratiquement pas de risque d'erreur de choix de la part de l'opérateur. Par contre la seconde méthode exige la lecture d'un symbole dans le menu (dans notre cas un chiffre mais dans le cas général ce peut être une lettre ou un groupe de caractères) et il est facile de commettre une erreur de lecture et taper un symbole à la place d'un autre.

Dans notre cas l'erreur n'est pas dramatique. Peu importe d'entendre "Au clair de la Lune" en lieu et place de "J'ai du bon tabac" ! Mais dans les programmes sérieux une erreur peut être une cause de perte de temps ou même être catastrophique, bien que dens un programme correctement conçu toute erreur de la part de l'opérateur doive être sans conséquence facheuse ou pouvoir être corrigée.

Si l'on tient à utiliser la seconde méthode, il est alors recommandé d'utiliser comme symboles repères, des symboles ayant une signification rappelant le "plat" à choisir dans le menu (initiales par exemple), plutot que des chiffres de 1 à N.
La détection du choix se fait alors par une variable Z bâtie à partir de variables logiques.

Supposons par exemple que nous ayons retenu les symboles suivants :

L T F P B C

Nous bâtirons la variable Z par la formule :

```
1-(R$="L")-2*(R$="T")-3*(R$="F")-4*(R$="P")-5*(R$="B")-6*(R$="C")
```

qui prend les valeurs 2, 3, 4, 5 , 6 ou 7 suivant que le symbole figurant dans R$ est L, T, F, P, B ou C et la valeur 1 en cas de choix d'une valeur non prévue.

Dans ces conditions le risque d'erreur est réduit, meis il reste de toute façon plus important qu'avec la méthode du crayon optique.

Nous allons examiner une troisième méthode qui, bien que n'utilisant pas la désignation directe par le crayon optique, n'impose pas à l'opérateur une lecture et une frappe de symbole au clavier.

Le principe consiste à faire défiler un "spot" représenté par un symbole devant les différents "plats" du menu, à l'aide des Flèches Heut et Bas et, si nécessaire avec les Flèches Gauche et Droite. L'opérateur amène ainsi le "spot" en face du "plat" choisi et précise son choix en tapant une touche, par exemple la touche ENTREE ou le touche ACC.

Nous allons modifier en conséquence le bloc logique débutant en 500.

Nous déplacerons par les Flèches Haut et Bas un "spot" qui sera centré dans l'une des "touches" carrées dessinées sur l'écran en face de chaque titre de chanson et numérotées de 1 à 5 ou dans la "touche" de cloture marquée C. Pour identifier la "touche" où est le spot, le symbole au centre du carré (chiffre de 1 à 5 ou lettre C) s'affichera en Rouge (1) sur fond de caractère Bleu (4).

A chaque déplacement du spot provoqué par la frappe de l'une des deux Flèches de déplacement vertical, le symbole à la dernière position du spot redeviendra Noir (0) sur fond Blanc (7) et le symbole à la nouvelle position passera au Rouge sur fond Bleu.

```
500 X=1:CC=37:LL=2:Z$="1":GOTO 900

510 R$=INPUT$(1):R=ASC(R$)

520 Z=1-(R=22)-2*(R=10)-3*(R=11)

530 ON Z GOTO 510,800,600,700

600 X=X+1:IF X=7 THEN X=1

610 GOTO 900

700 X=X-1:IF X=0 THEN X=6

710 GOTO 900

800 SCREEN 0,7,7

810 ON X GOTO 1000,2000,3000,4000,5000,6000

900 COLOR 0,7:LOCATE CC,LL,0:PRINT Z$

910 CC=1-36*(X(6):LL=-12*(X=6)-(5*X-3)*(X(6)

920 IF X=6 THEN Z$="C" ELSE Z$=CHR$(48+X)

930 LOCATE CC,LL,0:COLOR 1,4:PRINT Z$

940 GOTO 510
```

*** 500
Initialisation de la position du spot.

a/ X est un compteur variant de 1 à 6, repérant la position du spot. Pour X = 1 à 5 le spot est dans les "touches écran" 1 à 5. Pour X = 6 le spot est dans la touche "C". A l'initialisation X est égal à 1 (nous plaçons le spot dans la "touche" "1").

b/ CC est le numéro de colonne où est placé le spot, soit CC = 37 pour X de 1 à 5 et CC = 1 pour X = 6. A l'initialisation CC = 37.

c/ LL est le numéro de ligne où est placé le spot. A l'initialisation LL = 2.

d/ Z$ est le symbole à afficher au centre de la touche. A l'initialisation Z$ = "1".

e/ Branchement en 900 pour afficher la première position du spot (après chaque commande de déplacement par l'une des deux Flèches, le programme est branché sur 900).

*** 510
a/ Lecture de la touche du clavier frappée par l'opérateur -(M.Ref 174)-

b/ R est le numéro de code ASCII correspondant à R$.

*** 520
Calcul de la variable Z à partir des variables logiques, (R=22) (VRAI si la touche frappée est ACC, 22 étant le numéro de code ASCII de cette touche -(M.Ref 219)-), (R=10) (VRAI si la touche Flèche Bas a été tapée) et (R=11) (VRAI si la touche Flèche Haut a été tapée). Z prend les valeurs 2, 3 ou 4 suivant que l'opérateur a tapé la touche ACC, la Flèche Bas ou la Flèche Haut et la valeur 1 pour toute autre touche.

*** 530
Branchement :
en 510 pour Z = 1 (erreur de frappe, retour à une nouvelle lecture)
en 800 pour Z = 2 (touche ACC)
en 600 pour Z = 3 (Flèche Bas)
en 700 pour Z = 4 (Flèche Haut)

*** 600
La Flèche Bas a été frappée. Le compteur X est donc incrémenté de 1. Si X atteint la valeur 7 il est alors ramené à la valeur 1.

*** 610
Branchement en 900 pour l'"effacement" du spot à l'ancienne position et son affichage à la nouvelle.

*** 700
La Flèche Haut a été frappée. Le compteur X est donc décrementé de 1. Si X atteint la valeur 0, il est alors ramené à la valeur 6.

*** 710
Branchement en 900 pour l'"effacement" du spot à l'ancienne position et son affichage à la nouvelle.

*** 800
Les caractères seront Noirs (0) sur fond et entourage Blancs (7) pour l'affichage des titres.

*** 810
Suivant la valeur du compteur X, branchement aux étiquettes 1000 à 5000 (exécution de la musique) ou 6000 (clôture).

*** 900
a/ Fixe la couleur caractère au Noir (0) sur fond Blanc (7).

C'est avec cette couleur que sont affichés les chiffres et la lettre C dans les touches où ne se situe pas le "spot". L'affichage du caractère Z$ avec cette couleur "effacera" le spot.

b/ Positionne le curseur en colonne CC, ligne LL, CC et LL étant les valeurs calculées en fonction de la précédente valeur du compteur X. Ce sont donc les coordonnées de la précédente position du spot.

c/ Affiche le caractère Z$, c'est-à-dire le caractère correspondant à la dernière position du spot et calculé avec la précédente valeur du compteur X (Les valeurs de CC, LL et Z$ sont calculées en 910 en fonction de X. Au début du programme ces valeurs sont initialisées à 37, 2 et "1" en 500).

*** 910
a/ Calcul de CC en fonction de X. Pour X de 1 à 5, CC vaut 37, pour X = 6, CC vaut 1.

b/ Calcul de LL en fonction de X. Pour X de 1 à 5, LL vaut 5 * X - 3 et pour X = 6, LL vaut 12.

*** 920
Calcul du caractère Z$ en fonction de X. Pour X = 6, Z$ est le caractère "C" sinon pour X de 1 à 5, Z$ est CHR$(48+X), c'est-à-dire "1" pour X = 1,.., "5" pour X = 5.

*** 930
a/ Positionne le curseur en CC,LL calculés en 910

b/ Fixe la couleur caractère au Rouge (1) sur fond Bleu (4). C'est la couleur choisie pour faire apparaître le spot.

c/ Affiche le caractère Z$.

*** 940
Branche en 510 pour une nouvelle commande.

* * * * * *

Nous venons de voir trois méthodes de choix dans un menu. Il faut retenir que la méthode du crayon optique et la méthode de déplacement d'un spot sont les plus fiables et évitent au maximum les erreurs de l'opérateur. La méthode de choix par lecture et recopie est plus rapide à programmer, mais moins fiable et moins agréable pour l'opérateur.

3

Test musique

Nous nous proposons d'explorer les possibilités de l'instruction PLAY -(M.Ref 201)-, en particulier l'effet des "attributs", attaque, longueur, octave et tempo que l'on place en début des chaînes de notes. Nous allons pour cela bâtir un programme permettant de faire varier ces quatre paramètres et d'examiner les conséquences de ces variations sur la mélodie.

Nous essayerons l'effet des quatre paramètres sur une gamme chromatique montante et une gamme chromatique descendante séparées par une pause.

Les valeurs des quatre paramètres seront choisies en début d'essai puis les deux gammes jouées. Il sera possible alors soit de recommencer un essai soit de clôturer. Lorsqu'un nouvel essai est tenté, les valeurs des quatre paramètres de l'essai précédent sont affichés et l'opérateur peut, soit conserver certains paramètres, soit les modifier.

Les valeurs initiales des paramètres seront les valeurs par défaut, c'est à dire A = 0, L = 24, O = 4 et T = 5 -(M.Ref 201)-

Le programme se composera de trois parties, un "questionnaire" à remplir avec les valeurs désirées des quatre paramètres, un bloc où sera bâti la chaîne de caractères satisfaisant les réponses au "questionnaire" et où sera "joué" la mélodie correspondante, enfin un "menu" permettant à l'opérateur de poursuivre par un autre essai ou de clôturer.

Le premier bloc est le suivant :

```
10 M1$="DODO#RERE#MIFAFA#SOSO#LALA#SI":
   M2$="SILA#LASO#SOFA#FAMIRE#REDO#DO":EF$="     ":
   S$="O4DOO5DO":A=0:L=24:O=4:T=5

20 CLS:SCREEN 4,6,6

30 LOCATE 5,5:PRINT "OCTAVE   O (1 a 5)   ="

40 LOCATE 5,7:PRINT "LONGUEUR L (1 a 96)  ="

50 LOCATE 5,9:PRINT "ATTAQUE  A (0 a 255) ="

60 LOCATE 5,11:PRINT "TEMPO    T (1 a 255) ="

70 COLOR 1

80 LOCATE 0,5:PRINT USING "###";O

90 LOCATE 0,7:PRINT USING "###";L

100 LOCATE 0,9:PRINT USING "###";A

110 LOCATE 0,11:PRINT USING "###";T

120 COLOR 0

200 LOCATE 27,5:INPUT U$:IF U$="" GOTO 220

210 GOSUB 1000:IF E=1 OR VAL(U$)(1 OR VAL(U$))5
    THEN PLAY S$:LOCATE 27,5:PRINT EF$:
    GOTO 200 ELSE O=VAL(U$)

220 LOCATE 27,5:PRINT USING "###   ";O

230 LOCATE 27,7:INPUT L$:IF L$="" GOTO 250

240 GOSUB 1000:IF E=1 OR VAL(U$)(1 OR VAL(U$))96
    THEN PLAY S$:LOCATE 27,7:PRINT EF$:
    GOTO 230 ELSE L=VAL(U$)

250 LOCATE 27,7:PRINT USING "###   ";L

260 LOCATE 27,9:INPUT A$:IF A$="" GOTO 280
```

```
270 GOSUB 1000:IF E=1 OR VAL(U$)<0 OR VAL(U$))255
    THEN PLAY S$:LOCATE 27,9:PRINT EF$:
    GOTO 260 ELSE A=VAL(U$)

280 LOCATE 27,9:PRINT USING "###   ";A

290 LOCATE 27,11:INPUT T$:IF T$="" GOTO 310

300 GOSUB 1000:IF E=1 OR VAL(U$)<1 OR VAL(U$))255
    THEN PLAY S$:LOCATE 27,11:PRINT EF$:
    GOTO 290 ELSE T=VAL(U$)

310 LOCATE 27,11:PRINT USING "###   ";T

320 LOCATE 20,23



1000 E=0

1010 FOR I=1 TO LEN(U$)

1020 Z=ASC(MID$(U$,I,1))

1030 IF Z<48 OR Z>57 THEN E=1:RETURN

1040 NEXT:RETURN
```

*** 10

a/ et b/ La chaîne M1$ contient la gamme chromatique montante du DO au SI. La chaîne M2$ contient la gamme chromatique descendante du SI au DO. Nous serons amenés à compléter la gamme montante par un DO final à l'octave supérieure et la gamme descendante par un même DO initial. Mais il est nécessaire de ne les adjoindre qu'une fois choisie l'octave, car nous ne pouvons pas "fabriquer" un DO de l'octave 6. Nous verrons en 410 comment résoudre cette difficulté.

c/ La chaîne EF$ contient quatre espaces destinés à effacer les caractères à l'écran en cas d'erreur de frappe détectées par le sous programme 1000.

d/ la chaîne S$ est une chaîne de deux notes, un 00 de l'octave 4 et un 00 de l'octave 5, que nous utiliserons comme signal d'alarme en cas d'erreur détectée par le sous programme 1000.

e/ La valeur par défaut de A est 0 -(M.Ref 201)-

f/ La valeur par défaut de L est 24

g/ La valeur par défaut de O est 4

h/ La valeur par défaut de T est 5

*** 20

a/ Efface l'écran

b/ Les caractères seront Bleu (4) sur fond et entourage Cyan (6).

*** 30 - 60

Affiche les quatre lignes (de 5 à 11) du questionnaire. Les valeurs limites de chacun des quatre paramètres sont rappelées entre parenthèses.

*** 70

Passe la couleur des caractères au Rouge (1).

La couleur de fond de caractère reste au Cyan puisque le second paramètre de COLOR n'est pas précisé.-(M.Ref 189)-

*** 80 - 110

Affichage en colonne 0 des valeurs précédemment utilisées pour les quatre paramètres.

L'instruction PRINT USING -(M.Ref 90)- permet de justifier les valeurs sur la droite, avec les unités en colonne 2 (imposé par les 3 dièses de l'image utilisée).

Les valeurs sont affichées en Rouge imposé par l'instruction 70

*** 120

Passe la couleur des caractères au Noir (0).

La couleur de fond de caractère reste au Cyan puisque le second paramètre de COLOR n'est pas précisé.-(M.Ref 189)-

*** 200

a/ Place le Curseur en colonne 27, ligne 5, c'est à dire après le signe = de la question posée en ligne 5, à savoir quelle est la valeur du paramètre O.

b/ Affiche un point d'interrogation en colonne 27 et le curseur en colonne 30 puis met le programme en attente jusqu'à ce qu'une chaîne de caractères terminée par un "ENTREE" ait été tapée au clavier -(M.Ref 65)-.

La chaîne est stockée dans la varible chaîne U$.

c/ Si la chaîne U$ est une chaîne nulle, c'est à dire si la réponse de l'opérateur a été un simple "ENTREE", le programme saute l'instruction 210 et passe directement en 220.

Or en 210 le contenu de U$ est testé par le sous programme 1000, puis chargé dans la variable O après transformation en variable numérique par l'instruction VAL.

Autrement dit une réponse par "ENTREE" ne modifie pas la valeur du paramètre.

C'est pourquoi les valeurs précedentes de quatre paramètres sont affichées à gauche du questionnaire pour permettre à l'opérateur de ne modifier que les paramètres qui l'interessent et ceci de façon aisée en particulier sans avoir à répéter les valeurs qui n'ont pas à être modifiées.

On retiendra cette méthode de modification des paramètres à choisir par l'opérateur au début d'un programme.

Si de nombreux choix, avec variations de quelques paramètres seulement, sont à faire, la tâche de l'opérateur en est considérablement simplifiée. Le succès d'un programme dépend bien souvent de l'observation de principes aussi simples que celui-ci.

*** 210

a/ Branchement au sous programme 1000 pour vérifier que la chaîne entrée en U$ ne contient que des chiffres à l'exclusion de tout autre caractère. Si le test est négatif la variable E prend la valeur 1 (voir le sous programme 1000).

b/ Si E est égal à 1 (test 1000 négatif, c'est à dire chaîne U$ contenant des caractères qui ne sont pas des chiffres) ou si la valeur numérique représentée par la chaîne U$ et calculée par la fonction VAL -(M.Ref 151)- n'est pas dans la fourchette autorisée (1 à 5), les deux notes DO d'alarme sont joués (PLAY S$), quatre espaces d'effacement sont affichés à partir de la colonne 27 (LOCATE 27,5:PRINT EF$), ce qui efface le message erroné et le programme est rebranché en 200 pour une nouvelle réponse.

c/ Si les tests sont positifs (réponse plausible) (instruction ELSE-(M.Ref 61)-) la chaîne de caractères U$ est transformée en valeur numérique stockée dans la variable O -(M.Ref 151)-.

*** 220

a/ Place le curseur en colonne 27, ligne 5 (là où est actuellement affiché le point d'interrogation).

b/ Affiche la valeur de O retenue en plaçant le chiffre des unités en colonne 29. On notera que l'image utilisée dans l'instruction PRINT USING comporte les 3 signes dièses, suivis de trois espaces qui sont donc affichés dans les colonnes 30 à 32, effaçant les caractères qui s'y trouvaient éventuellement suite à l'introduction de données par l'instruction INPUT.

*** 230 - 310

Ces instructions sont analogues aux trois instructions précédentes. Elles permettent de recueillir les valeurs des trois autres paramètres L, A et T après avoir vérifié leur vraissemblance.

*** 320

Place le curseur en colonne 20, ligne 23. On constatera que le curseur clignote en suivant la mesure de la musique.

*** 1000

Ouverture du sous programme de test de la chaîne U$. La variable E de signal d'erreur est mise à zéro (pas d'erreur).

*** 1010

Ouverture d'une boucle en I, I étant le compteur de caractères de la chaîne U$. On fait donc varier I de 1 à la longueur de U$ calculée par LEN(U$).

*** 1020

Z est le numéro de code ASCII du caractère de rang I de U$, ce ceractère étent extrait de U$ par la fonction MID$ -(M.Ref 130)-

*** 1030

Si le numéro de code ASCII Z est inférieur à 48 (code de "0") et supérieur à 57 (code de "9") la variable E est posée égale à 1 (signel d'erreur puisque le caractère testé n'est pas un chiffre) et la sortie du sous programme est provoquée par l'instruction RETURN. Il est en effet inutile de poursuivre le test une fois une première erreur détectée.

*** 1040

a/ Fermeture de la boucle en I.

Il est inutile de préciser la variable sur laquelle on boucle puisqu'il n'y a qu'une seule boucle en service.

b/ Instruction de fin de sous programme -(M.Ref 57)-

* * * * * *

Le bloc de calcul de la chaîne à jouer par PLAY est le suivant :

```
400 O$="O"+CHR$(48+O):
    L$=STR$(L):L$="L"+RIGHT$(L$,LEN(L$)-1):
    A$=STR$(A):A$="A"+RIGHT$(A$,LEN(A$)-1):
    T$=STR$(T):T$="T"+RIGHT$(T$,LEN(T$)-1)

410 DOF$="":IF O<5 THEN DOF$="O"+CHR$(49+O)+"DO"

420 PLAY O$+L$+A$+T$+M1$+DOF$+"L96P"+
        L$+DOF$+O$+M2$
```

*** 400

Il s'agit de fabriquer quatre chaînes présisant les quatre commendes d'octave (O), de durée de note (L), d'attaque (A) et de tempo (T) concaténant une lettre (O, L, A, T) et les valeurs numériques choisies, transformées en chaînes de caractères.

a/ La valeur numérique O est transformée en caractère par CHR$(48+O) (48 est le numéro de code ASCII de zéro, ainsi pour O=5 par exemple le caractére est "5" (code 48+5=53). O$ est alors le résultat de la concaténetion du caractère "O" et de CHR$(48+O). O$ est alors la chaîne de commande de niveau d'octave.

b/ et c/ La valeur numérique L est transformée en chaîne L$ per la fonction STR$ -(M.Ref 147)-. La chaîne qui en résulte débute par un espace qui est la place réservée au signe (un espace à la place du signe + ou signe -). Oans notre cas la variable transformée est positive, donc la chaîne commence toujours par un espace. Avant de concaténer cette chaîne derrière le caractère "L", il est nécessaire de supprimer cet espace. C'est le but de l'instruction RIGHT$ -(M.Ref 136)- qui extrait la partie droite de L$ sur une longueur égale à la longueur de L$ (LEN -(M.Ref 128)-) moins un. On obtient en définitive la chaîne de commande L$ de durée des notes.

Les instructions suivantes sont analogues et permettent de calculer les chaînes de commandes d'attaque A$ et de tempo T$.

*** 410
Il nous faut calculer maintenant la chaîne correspondant eu DO de l'octave supérieure. Cette note ne peut être calculée que si le niveau d'octave choisi est inférieur au niveau 5, car il n'y e pes de OO d'octave 6. Nous posons donc le chaîne DOF$ (mnémonique pour OO final) égale à une chaîne nulle e priori et si O est inférieur à 5, OOF$ est obtenu en concaténant le caractère "O", le caractère CHR$(49+O), qui est le niveau d'octave choisi augmenté de un (49 est le numéro de code ASCII de 1) et la chaîne "OO".

*** 420
La chaîne de caractères suivant l'instruction PLAY est composée :

- des quatre chaînes de commandes O$, L$, A$, T$ imposent le niveau d'octave, la durée des notes, l'attaque et le tempo,

- de la chaîne M1$, gamme chrometique montante,

- de la chaîne DOF$, DO de l'octave supérieure s'il existe,

- de la chaîne "L96P" qui provoque un pause, entre les deux gammes, de durée égale à 96,

- de la chaîne de commande L$ qui rétablit la durée initiale des notes, modifiée par la commande "L96" précédant la pause,

- de la chaîne DOF$, DO de l'octave supérieure s'il existe,
pour débuter la gamme descendante,

- de la chaîne de commende O$ qui rétablit le niveau d'octave, modifié par la chaîne DOF$,

- de la chaîne M2$, gamme chromatique descendante.

* * * * * *

Il nous reste maintenant à examiner le dernier bloc, destiné à proposer un "menu" à l'opérateur lui permettant de choisir entre la poursuite des essais ou la clôture.

```
500 CLS:SCREEN 6,4,4:LOCATE 5,10:PRINT "Pour"

510 LOCATE 5,13:PRINT "Un autre essai Tapez ";:
    COLOR 0,3:PRINT "ENTREE"

520 LOCATE 5,15:PRINT "Cloturer        Tapez   ";:
    COLOR 0,6:PRINT "C"

530 R$=INPUT$(1):ON 1-(R$="C")-2*(ASC(R$)=13)
    GOTO 530,600,20

600 CLS:SCREEN 4,6,6:END
```

*** 500
a/ Efface l'écran.

b/ Fixe la couleur des caractères au Cyan (6) sur fond et entourage Bleu (4).

c/ et d/ Affiche le message "Pour" en colonne 5, ligne 10.

*** 510 - 520
Affiche les deux choix proposés à l'opérateur, poursuite ou clôture.

On notera la façon dont sont représentées les deux touches de commandes retenues pour le choix, caractères Noirs (0) sur fond Jaune (3), pour la touche "ENTREE", caractères Noirs (0) sur fond Cyan (6), pour la touche "C", de façon à simuler la couleur et la forme des deux touches du clavier à utiliser.

Là encore on peut considérer ces détails comme peu importants. Il font pourtant toute la différence entre un programme réberbatif et un programme agréable à utiliser.

*** 530
a/ Recueil de la réponse R$ de l'opérateur.

b/ Suivant que la touche frappée par l'opérateur est une touche quelconque, la touche "C" ou la touche "ENTREE", branchement en 530 pour une nouvelle réponse, en 600 pour la clôture, en 20 pour un nouvel essai.

*** 600
a/ Efface l'écran.

b/ Rétablit les couleurs normales écran pour une utilisation ultérieure.

c/ L'instruction END est indispensable pour éviter que le programme ne continue en séquence et n'attaque les instructions 1000 et suivantes du sous programme test, provoquent le message d'erreur RG (RETURN without GOSUB, RETURN sans avoir rencontré d'instruction GOSUB).

Nous laissons le lecteur utiliser le programme pour apprécier de lui même l'effet des paramètres d'attaque et de tempo plus subtil que celui des paramètres durée ou niveau de l'octave.

Il constatera qu'il n'y e pas intéret à s'éloigner beaucoup des valeurs par défeut, 0 pour l'attaque et 5 pour le tempo sous peine d'obtenir des effets curieux sinon inesthétiques !

Le progremme définitif est donc le suivant :

```
10 M1$="DODO#RERE#MIFAFA#SOSO#LALA#SI":
   M2$="SILA#LASO#SOFA#FAMIRE#REDO#DO":EF$="     ":
   S$="O4DOOSDO":A=0:L=24:O=4:T=5

20 CLS:SCREEN 4,6,6

30 LOCATE 5,5:PRINT "OCTAVE   O (1 a 5)   ="

40 LOCATE 5,7:PRINT "LONGUEUR L (1 a 96)  ="

50 LOCATE 5,9:PRINT "ATTAQUE  A (0 a 255) ="

60 LOCATE 5,11:PRINT "TEMPO    T (1 a 255) ="

70 COLOR 1

80 LOCATE 0,5:PRINT USING "###";O

90 LOCATE 0,7:PRINT USING "###";L

100 LOCATE 0,9:PRINT USING "###";A

110 LOCATE 0,11:PRINT USING "###";T

120 COLOR 0

200 LOCATE 27,5:INPUT U$:IF U$="" GOTO 220

210 GOSUB 1000:IF E=1 OR VAL(U$)<1 OR VAL(U$)>5
    THEN PLAY S$:LOCATE 27,5:PRINT EF$:
    GOTO 200 ELSE O=VAL(U$)

220 LOCATE 27,5:PRINT USING "###   ";O

230 LOCATE 27,7:INPUT L$:IF L$="" GOTO 250

240 GOSUB 1000:IF E=1 OR VAL(U$)<1 OR VAL(U$)>96
    THEN PLAY S$:LOCATE 27,7:PRINT EF$:
    GOTO 230 ELSE L=VAL(U$)

250 LOCATE 27,7:PRINT USING "###   ";L

260 LOCATE 27,9:INPUT A$:IF A$="" GOTO 280

270 GOSUB 1000:IF E=1 OR VAL(U$)<0 OR VAL(U$)>255
    THEN PLAY S$:LOCATE 27,9:PRINT EF$:
    GOTO 260 ELSE A=VAL(U$)
```

```
280 LOCATE 27,9:PRINT USING "###    ";A

290 LOCATE 27,11:INPUT T$:IF T$="" GOTO 310

300 GOSUB 1000:IF E=1 OR VAL(U$)<1 OR VAL(U$)>255
    THEN PLAY S$:LOCATE 27,11:PRINT EF$:
    GOTO 290 ELSE T=VAL(U$)

310 LOCATE 27,11:PRINT USING "###    ";T

320 LOCATE 20,23

400 O$="O"+CHR$(48+O):
    L$=STR$(L):L$="L"+RIGHT$(L$,LEN(L$)-1):
    A$=STR$(A):A$="A"+RIGHT$(A$,LEN(A$)-1):
    T$=STR$(T):T$="T"+RIGHT$(T$,LEN(T$)-1)

410 DOF$="":IF O<5 THEN DOF$="O"+CHR$(49+O)+"DO"

420 PLAY O$+L$+A$+T$+M1$+DOF$+"L96P"+
        L$+DOF$+O$+M2$

500 CLS:SCREEN 6,4,4:LOCATE 5,10:PRINT "Pour"

510 LOCATE 5,13:PRINT "Un autre essai Tapez ";:
    COLOR 0,3:PRINT "ENTREE"

520 LOCATE 5,15:PRINT "Cloturer        Tapez    ";:
    COLOR 0,6:PRINT "C"

530 R$=INPUT$(1):ON 1-(R$="C")-2*(ASC(R$)=13)
    GOTO 530,600,20

600 CLS:SCREEN 4,6,6:END

1000 E=0

1010 FOR I=1 TO LEN(U$)

1020 Z=ASC(MID$(U$,I,1))

1030 IF Z<48 OR Z>57 THEN E=1:RETURN

1040 NEXT:RETURN
```

4

Mire

Avant d'aborder des programmes plus ambitieux, exécutons un programme simple à écrire, encore qu'il ne recèle de nombreux pièges, mais aisés à detecter après essai. Nous proposons de bâtir une mire analogue aux mires de télévision. Cela aura au moins le mérite de fournir un outil de test pour notre récepteur vidéo.

Nous recommandons au lecteur, avant d'eborder cet exercice, de lire soigneusement l'Annexe I consecrée à la gestion des couleurs, car les notions qui y sont exposées sont indispensables à la compréhension de la programmation.

La mire sera essentiellement composée d'un cadre rectangulaire Rouge entourant tout l'écran, d'une grille Bleu, constituée de 19 lignes verticales et de 11 lignes horizontales et d'une plage circulaire Bleu au centre de l'écran.

A la partie supérieure de la plage circulaire, nous inscrirons MIRE et TO7 en caractères double taille. A la partie inférieure, nous placerons les quatre flèches verticales et horizontales que nous avons appris à tracer dans le premier exercice, toujours en taille double (pourquoi pas !).

Au centre de la plage circulaire Bleu, nous tracerons huit plages carrées, avec les huit couleurs disponibles, en plaçant en leur centre le numéro correspondant à chaque couleur, chaque chiffre ayant lui-même la couleur du carré suivant.

Enfin nous disposerons sur les bords de la grille, toujours en caractères double taille, les 10 chiffres et les 26 lettres de l'alphabet.

Partons de cette idée générale de construction de la mire et c'est en "cours de route" que nous serons capables de préciser les diverses positions des symboles et des graphismes.

* * * * * *

Attaquons tout d'abord la construction du cadre Rouge, de la grille Bleu et de la plage circulaire Bleu.

```
10 CLS:SCREEN 4,6,6:LOCATE 0,0,0

20 X0=160:Y0=96

30 BOX (0,0)-(319,192),1

40 BOXF (X0-80,Y0-80)-(X0+80,Y0+80),6

50 FOR Y=16 TO 176 STEP 16:LINE (0,Y)-(319,Y):NEXT

60 FOR X=16 TO 64 STEP 16:LINE (X,1)-(X,191):NEXT

70 FOR X=80 TO 224 STEP 16:
   LINE (X,1)-(X,Y0-80):
   LINE (X,Y0-79)-(X,Y0+80),-5:
   LINE (X,Y0+81)-(X,191):NEXT

80 FOR X=240 TO 304 STEP 16:
   LINE (X,1)-(X,191):NEXT

100 R=78:P=57.3

110 FOR T=1 TO 45 STEP 2

120  TT=T/P:C=COS(TT):S=SIN(TT)

130  X1=X0+R*C:Y1=Y0-R*S:X2=X0-R*C:Y2=Y0+R*S

140  X3=X0+R*S:Y3=Y0-R*C:X4=X0-R*S:Y4=Y0+R*C

150  BOXF (X1,Y1)-(X2,Y2),-5:
     BOXF (X3,Y3)-(X4,Y4),-5

160 NEXT
```

*** 10

a/ Effece l'écran.

b/ Fixe la couleur des caractères qui seront Bleu (4) sur fond et entourage Cyan (6).

c/ Place le curseur en colonne 0, ligne 0 et l'efface.

*** 20

Oéfinit les deux constantes XO et YO, coordonnées du centre de l'écran

*** 30

Trace un cadre Rouge (1) avec le coin haut-gauche en 0,0 et le coin bas-droite en 319,192.

La distance entre les traits constituant la grille étant de 16 points, il est possible de tracer 12 "ceses" de 16 points de heut dans le sens verticel. Oans ces conditions si le premier trait est à l'ordonnée 0, le dernier trait est à l'ordonnée 12 x 16 = 192, ce qui explique l'ordonnée du coin bes-droit de notre cadre.

De même il est possible de trecer 20 "cases" de 16 points de large dans le sens horizontal. Si le premier trait vertical est à l'abscisse 0, le dernier treit verticel est alors à l'abscisse 16 x 20 = 320. Nous avons triché d'un point en mettant le bord vertical droit en 319 (il n'y e pas de point 320 !).

*** 40

Nous avons l'intention de placer une plage circulaire au centre de l'écran et d'y tracer des symboles et des graphismes.

Or on sait (voir Annexe I) que l'on a intérêt à tracer, en couleur de fond, les plages sur lesquelles on doit dessiner. La plage circulaire sera donc tracée en Bleu fond (-5). Mais l'ensemble de l'écran est obligatoirement en couleur fond (imposée par l'instruction SCREEN en 10). Pour éviter que le cercle Bleu (fond) ne vienne "baver" sur le fond général (ce qui se traduireit par un "cercle" avec des crans), il est nécessaire de trecer eutour du futur cercle une plage cerrée en couleur carectère Cyan (6). Cette plage ne sera pas visible sur l'écran car tracée en couleur carectère Cyan sur couleur fond Cyan. Mais elle est indispensable pour éviter les "bavures".

Pour s'en bien convaincre il suffit d'essayer de tracer le cercle avec les instructions 100 à 160 en sautant les instructions 40 à 80 par une instruction

35 GOTO 100

Une fois l'effet de cette instruction essayé, effaçons-la et revenons à la construction du cercle.

Le cercle sera centré sur le point de coordonnées XO,YO et son rayon sera de 78 points.Cette dernière valeur a été choisie expérimentalement au voisinage de 80, de façon à obtenir une figure se rapprochant le plus possible d'un cercle avec le moins de "merches" possible. Dans ces conditions il est nécessaire de tracer notre plage en couleur caractère Cyan de telle sorte que le cercle se situe en entier à l'intérieur. Ceci explique les valeurs 80 figurant dans l'instruction 40. Le lecteur aura intérêt à faire varier les valeurs du rayon et de la plage pour bien voir l'effet de ces paramètres.

*** 50

a/ ouverture d'une boucle en Y variant de 16 à 176 avec un pas de 16, permettant de tracer les 11 lignes horizontales Bleu de la grille.

b/ Tracé de la ligne horizontale d'ordonnée Y entre les points d'abscisses 0 et 319.

La couleur de la ligne est le Bleu; la valeur du paramètre couleur n'ayant pes été précisée, c'est la couleur caractère définie per l'instruction SCREEN qui est retenue.

On aurait pu envisager de ne tracer la ligne horizontele qu'entre les points 1 et 318 pour éviter de faire déborder les horizontales Bleu sur le cadre Rouge. Le lecteur pourra vérifier que l'instruction LINE (1,Y)-(318,Y) ne modifie pas le résultat. En effet les points d'abscisses 1 à 7 étant colorés en couleur caractère Bleu, imposent la couleur Bleu au point d'abscisse 0 qui appartient au même bloc de huit points.

Si l'on tient à ne pes déborder sur le cadre Rouge, il est nécesseire de trecer les horizontales en couleur fond Bleu, et non pas en couleur cerectère, entre les points 1 et 318.

Il nous faut alors écrire l'instruction 50 sous la forme :

```
50 FOR Y=16 TO 176 STEP 16:LINE (0,Y)-(319,Y),-5:NEXT
```

Il faut noter que cette dernière solution n'est acceptable que parce que nous traçons une ligne horizontale.

Dans ce seul cas en effet les "bavures" d'un ligne tracée, couleur fond sur couleur fond, ne se voient pas.

Le lecteur pour s'en convaincre n'a qu'à essayer de tracer une droite de 0,0 à 319,199, par exemple en couleur fond sur couleur fond.

Tapons les trois instructions suivantes et essayons les en faisant RUN 1000.

```
1000 CLS:SCREEN 4,6,6

1010 Y1=0:Y2=199

1020 LINE (0,Y1)-(319,Y2),-5
```

Si par contre nous supprimons le paramètre de couleur -5, nous obtenons un résultat correct sans "bavures" (mais non sans "escaliers"). De même les "bavures" sont sans effet sur le graphisme pour une horizontale (avec Y2 = 0 par exemple).

N'oublions pas d'effacer ces trois étiquettes par DELETE 1000-1030 après essai !

Le lecteur choisira la solution qu'il préfère !

c/ Fermeture de la boucle en Y

*** 60 - 80

Il s'agit de tracer maintenant les 19 verticales Bleu, de 16 en 16, depuis l'abscisse 16 jusqu'à l'abscisse 304. La programmation ne sera pas la même suivant que nous serons sur un fond en couleur type fond ou sur un fond en couleur type caractère. Il est en effet nécessaire, comme nous venons de le montrer ci-dessus, de tracer les lignes en couleur de type différent du "fond" sur lesquelles elles sont placées.

Pour les abscisses X de 16 à 64 et de 240 à 304, le fond, quelle que soit l'ordonnée Y, est de couleur type fond. Par contre pour les abscisses X, comprises entre 80 et 224, le fond est en couleur type caractère pour les ordonnées Y comprises entre YO - 79 et YO + 80, et en couleur type fond à l'extérieur de cet intervalle. Ceci explique les instructions sous les étiquettes 60 et 80, où les verticales sont tracées en couleur caractère Bleu (valeur 4 prise par défaut).

Par contre à l'étiquette 70, il est nécessaire de faire la distinction entre les verticales dans la plage YO - 79 et YO + 80, à tracer en couleur Bleu fond (-5), et celle à l'extérieur de la plage, à tracer en couleur Bleu caractère (4), c'est à dire sans préciser le paramètre couleur, la valeur 4 étant retenue par défaut.

* * * * * *

Parvenu à ce stade de la programmation, il est utile de tester les instructions que l'on vient d'écrire et de s'amuser (!) à faire varier quelques paramètres pour juger de l'effet produit, ce qui ne peut manquer d'être instructif.

Que le lecteur ne s'y trompe pas ! Les instructions 10 à 80 sont évidentes à justifier, une fois qu'elles ont été écrites et essayées. Il est infiniement moins évident de les écrire, a priori, sans erreurs. Ce n'est qu'après essai que l'on peut s'écrier "Bien sûr ! Cette instruction est erronnée" (en général le programmeur utilise un langage moins châtié). L'auteur lui-même n'est parvenu au résultat "évident" qu'après de nombreux tatonnements. Mais c'est là l'avantage du Micro. On peut essayer de nombreuses fois, car l'on dispose de la réponse immédiatement. Ce n'est toutefois pas une raison pour justifier le "bidouillage" (que l'Académie nous pardonne ce terme de métier) et faire n'importe quoi en espérant tomber par hasard sur la bonne solution !

Après ces considérations pratiques, attaquons la construction de la plage circulaire.

Rappelons quelques données mathématiques pour ceux qui ne manipulent pas les équations tous les jours.

Appelons C le centre du cercle (coordonnées p,q dans un système d'axes Ox,Oy), R son rayon, A le point à l'extrémité droite du rayon horizontal (coordonnées p+R,q), M un point du cercle et t l'angle ACM. On sait que les coordonnées x et y du point M sont :

$$x = p + R \cdot \cos t$$

$$y = q + R \cdot \sin t$$

Il nous faut transposer ces équations aux coordonnées écran en se souvenant

a/ que les angles doivent être exprimés en radians

b/ que las ordonnées écran Y croissent vers le bas de l'écran, donc en sens contraire de l'ordonnée y dans le systéme xOy.

*** 100
a/ R est le rayon du cercle

b/ P est le rapport 180/3.14159 permettant de transformer des degrés en radians.

Rappelons qu'il est préférable d'exprimar les constantes sous forme d'une variable alphanumérique plutôt que sous forme d'une chaîne de chiffres, surtout à l'intérieur d'une boucle, car on évite ainsi à l'interprèteur de transcrire la chaîne de chiffres en valeur numérique à chaque passage sur l'instruction.

*** 110
Ouverture d'une boucle en T (angle t des équations du cercle). On fait varier T de 1 à 45 degrés.

Nous allons en effet tracer une première plage rectangulaire colorée en Bleu (fond, -5),avec un coin en M et un deuxième coin en N symétrique de M par rapport au centre C.

Une deuxième plage rectangulaire de même couleur, aura un coin en M', symétrique de M par rapport à la première bissectrice des axes Cx et Cy, et un deuxième coin en N', symétrique de M' par rapport au centre C.

Si t est l'angle repérant M, l'angle t' repérant M' est égal à 90 - t. L'angle u repérant N est 180 + t et l'angle u' repérant N' est 180 + t'.

Ceci explique que nous n'ayons à faire varier T que de 1 à 45 degrés seulement, tout en obtenant la totalité de la plage circulaire, grâce au tracé des deux plages rectangulaires, la première correspondant aux angles de 1 à 45 degrés, la seconde correspondant aux angles de 89 à 45 degrés.

Rappelons enfin les formules classiques de trigonométrie :

cos (90 - t) = sin t sin (90 - t) = cos t

cos (180 + t) = - cos t sin (180 + t) = - sin t

On a enfin fait varier T avec un pas de 2.

Il est en effet inutile de calculer les points M de degrés en degrés étant donné la définition de l'écran.

On constate expérimentalement que la figure obtenue avec un pas de 2 ne présente pas plus d'"escaliers" qu'avec un pas de 1 et le temps de tracé est divisé par deux. Par contre un pas supérieur à 2 conduit à une figure trop dentelée.

*** 120
a/ TT est la valeur de T, exprimée en radians.

b/ et c/ C et S sont le cosinus et le sinus de TT.

L'utilisation de ces variables intervient dans le calcul des coordonnées des quatre points M, M', N et N'.

Il est inutile de calculer les valeurs du sinus et du cosinus des angles repérant les trois points M', N et N' puisque ces valeurs se déduisent des valeurs du sinus et du cosinus de t comme nous l'avons rappelé plus haut.

N'oublions pas que le calcul d'une ligne trigonométrique prend un temps non négligeable et que le tracé du cercle serait retardé notablement si nous calculions les sinus et cosinus des quatre angles.

*** 130
Calcul des coordonnées X1, Y1 de M, X2, Y2 de N.

On notera le signe - figurant dans la formule donnant Y1 (Y1=Y0-R*S) pour tenir compte du sens de variation des Y sur l'écran.

*** 140
Calcul des coordonnées X3, Y3 de M', X4, Y4 de N'.

*** 150
Tracé des deux plages rectangulaires en Bleu fond -5.

*** 160
Fermeture de la boucle en T

* * * * * *

Après avoir testé cet ensemble d'instructions, qui constitue un "bloc" exécutable, nous allons inscrire les divers symboles à l'intérieur de la plage circulaire Bleu fond et placer les huit plages colorées.

```
200 ATTRB 1,1:COLOR 7,4:LOCATE 16,4:PRINT "MIRE"

210 LOCATE 16,6:PRINT "TO 7"

220 A$=CHR$(22):COLOR 5,4:LOCATE 16,20:
    PRINT A$;",";A$;"-";A$;".";A$;"/":ATTRB 0,0

300 BOXF (112,64)-(215,127),4:
    BOX (103,63)-(216,128),1

310 FOR J=0 TO 1

320  FOR I=0 TO 3

330  X=113+24*I:C=15+3*I:U=I+4*J:CC=1+U:CCN=-CC:
     IF CC=8 THEN CC=0

340  BOXF (X,70+30*J)-(X+22,92+30*J),CCN

350  LOCATE C,10+4*J:COLOR CC,U:PRINT CHR$(48+U)
```

```
360 NEXT I

370 NEXT J

380 BOX (112,100)-(135,122),7
```

*** 200
a/ Fixe la taille des caractères à la taille double.

b/ Fixe la couleur des caractères au Blanc (7) sur fond Bleu (4).

c/ et d/ Affiche la chaîne "MIRE" en plaçant le coin bas-gauche du M (taille double) en colonne 16, ligne 4

*** 210
Affiche la chaîne "TO 7" avec le coin bas-gauche du T (taille double) en colonne 16, ligne 6.

*** 220
a/ La variable A$ est le caractère controle de numéro de code ASCII 22, c'est à dire le caractère correspondant à la touche ACC. On a vu -(M.Bas 1)- que ce caractère placé devant certains autres caractères permettait d'afficher des caractères spéciaux, en particulier les quatre flèches.

b/ Fixe la couleur des caractères au Magenta (5) sur fond Bleu (4).

c/ Place le curseur en colonne 16, ligne 20

d/ Affiche les quatre flèches obtenues en concatánant le caractère A$ (touche ACC) avec les caractères ",", "-", "." et "/". On remarquera que la concaténation a été commandée avec des point-virgules (;) et non avec le signe plus (+). Ceci est possible après la commande PRINT, la concaténation n'étant pas calculée mais seulement exécutée en affichant les caractères successivement sur l'écran.

*** 300
a/ Nous allons tracer huit plages rectangulaires avec les huit couleurs disponibles en plaçant en leur centre un chiffre représentant le numéro correspondant de la couleur. Ce chiffre sera affiché, comme tout caractère, en couleur caractère. Il est donc nécessaire que la couleur de la plage soit du type fond.

Or la plage circulaire Bleu a déjà été tracée en couleur fond. Il importe donc de placer autour de la zone que devront occuper nos huit plages colorées, une plage de couleur Bleu, mais de type caractère. C'est l'objet de la première instruction de 300.

b/ Pour délimiter, de façon "artistique" la zone des huit plages colorées, nous plaçons un cadre Rouge (1) défini par la seconde instruction.

*** 310
Ouverture d'une boucle en J.

J varie de 0 à 1.

J est le numéro de la "ligne" des quatre plages colorées.

*** 320
Ouverture d'une boucle en I.

I varie de 0 à 3.

I est le numéro de "colonne" de l'une des quatre plages colorées sur une même ligne.

*** 330
a/ X est l'abscisse du bord gauche de la plage de rang I.

Chaque plage a 22 points de large et est séparée de la plage suivante par 2 points (22 + 2 = 24).

b/ C est le numéro de colonne écran du centre de la plage de rang I où sera placé le numéro de la couleur de la plage.

c/ U est le numéro de la plage (0 pour la première en haut à gauche, 7 pour la dernière en bas à droite). La plage 0 sera Noire fond (-1), la dernière Blanche fond (-8).

d/ et e/ CC est la couleur caractère de la plage suivant la plage de rang U et CCN la couleur fond de la plage de rang U.

Pour I=0 et J=0, première plage en haut et à gauche (U = 0), CCN est égal à -1 (Noir fond) et CC est égal à 1 (Rouge caractère); pour I=2 et J=0 (plage suivante, U = 1), CCN est égal à -2 (Rouge fond) et CC est égal à 2 (Vert caractère)

f/ Si CC calculé est égal à huit (dernière plage,U = 7), il est nécessaire de ramener cette valeur à 0, numéro de couleur caractère du Noir. La plage "suivante", qui est la première (U = 0), est en effet de couleur Noir fond, comme nous venons de le voir en (e) ci dessus.

*** 340

Tracé de la plage colorée rectangulaire en couleur fond CCN, de largeur 22 points et de hauteur 22 points. La ligne graphique supérieure est d'ordonnée Y = 70 pour la rangée supérieure (J = 0) et Y = 100 pour la rangée inférieure (J = 1).

*** 350

a/ place le curseur au centre de la plage de rang U, en colonne C (calculé en 330,b) et en ligne 10 pour la première rangée (J = 0) ou en ligne 14 pour la seconde (J = 1).

b/ Fixe la couleur de caractère à CC (calculée en 330,d) et la couleur de fond de caractère à U (calculé en 330,c). Rappelons que la couleur de fond de caractère dans une instruction COLOR est fixée par le second paramètre exprimé avec un numéro de couleur positif. Pour la première plage (U = 0, couleur Noir fond CCN= -1), le chiffre représentant le numéro de couleur de la plage (0 en l'occurence) doit être tracé en couleur caractère Rouge (CC= 1) sur fond de caractère Noir (U = 0).

b/ 48 + U est le numéro de code ASCII du chiffre U. On affiche ainsi le chiffre 0 dans la première plage et le chiffre 7 dans la dernière. Pour la fonction CHR$ voir -(M.Ref 113)-

*** 360 - 370

Fermeture des deux boucles en I et J.

*** 380

La première case de la deuxième rangée (U = 4) de couleur fond Bleu (-5), n'apparait pas sur le fond général Bleu (Couleur caractère 4 rappelons-le). Il est donc plus esthétique de la faire apparaitre en l'entourant d'un cadre de couleur Magenta (5).

* * * * * *

Il nous reste maintenant, après avoir testé le dernier bloc exécutable, à tracer, en caractères double taille, les vingt six majuscules de l'alphabet, réparties sur les première et dernière "lignes" de la grille et les dix chiffres sur les première et dernière "colonnes"

```
400 ATTRB 1,1:COLOR 4,6

410 FOR I=0 TO 19:LOCATE 2*I,1:
    PRINT CHR$(65+I):NEXT

420 FOR I=0 TO 5:LOCATE 2*I+14,23:
    PRINT CHR$(85+I):NEXT

430 FOR I=0 TO 4:LOCATE 0,2*I+9:
    PRINT CHR$(48+I):NEXT

440 FOR I=5 TO 9:LOCATE 38,2*(I-5)+9:
    PRINT CHR$(48+I):NEXT

450 BOX (0,0)-(319,192),4:LINE (0,176)-(319,176)

460 LINE (0,16)-(319,16):LINE (0,176)-(319,176)

470 FOR X=16 TO 304 STEP 16:
    LINE (X,0)-(X,16):LINE (X,176)-(X,192):NEXT

480 FOR Y=64 TO 128 STEP 16:
    LINE (0,Y)-(16,Y):LINE (304,Y)-(319,Y):NEXT

490 LINE (304,0)-(304,192)
```

*** 400

a/ Fixe la taille des caractères à la taille double.

b/ Fixe la couleur des caractères au Bleu (4) sur Cyan (6).

*** 410

Affiche les 20 premières majuscules sur la première ligne de la grille.

Rappelons que 65 est le numéro de code ASCII de A; ainsi CHR$(65+I) est la Ième majuscule.

*** 420 - 440
Instructions analogues à la précédente pour l'affichage des 6 dernières majuscules et des 10 chiffres.

On remarquera que le fait d'afficher un caractère taille double dans une case de la grille, efface les lignes à gauche et au dessus du caractère.

Pour les caractères affichés sur la colonne de droite de la grille, il y a également effacement de la ligne verticale à droite.

En effet un caractere taille double occupe daux blocs de huit points. Les abscisses des premiers points des premiers blocs sont des multiples de huit et les ordonnées de la ligne supérieure sont également des multiples de huit. Ceci explique l'effacement des lignes gauche et supérieure de la portion de grille autour d'un caractère puisque ces lignes ont été tracées avec de telles coordonnées.

Enfin la dernière ligne verticale à droite de la grille a une abscisse de 319 (rappelons que nous avons dû tricher puisqu'il n'y a pas de points d'abscisse 320). Les points de cette verticale, au niveau d'un caractère tracé sur la dernière colonne, appartiennent donc au bloc caractère et sont par suite effacés à l'affichage du caractère.

Il nous faut donc rétablir la grille là où elle a été effacée.

*** 450
Rétablit le cadre extérieur. On remarquera que la couleur en est Bleu (4). Il est en effet impossible de tracer les parties verticales du cadre en une autre couleur sous peine de voir la moitié gauche des caractères à gauche et la moitié droite des caractères à droite changer de couleur.

*** 460 - 490
Ces instructions, de forme analogue à celle des instructions 50 à 80, rétablissent les portions de grille effacées par l'écriture des caractères.

* * * * * *

Après avoir testé ces dernières instructions constituant un bloc exécutable, il nous faut terminer le programme par quelques instructions permettant à l'opérateur de faire disparaitre la mire.

Nous ferons apperaitre sur la mire, le message "Tapez ENTREE", en affichant "ENTREE" avec les couleurs de la touche, c'est à dire caractères Noir (0) sur fond Jaune (3).

Sur commande de la touche "ENTREE" par l'opérateur, détectée par une instruction INPUT$(1), nous ferons disparaitre la mire en la faisant se déplacer lentement vers le haut, grâce à l'instruction CONSOLE.

```
500 ATTRB 0,0:LOCATE 14,17:COLOR 1,4:
    PRINT "Tapez ";:COLOR 0,3:PRINT "ENTREE"

510 R$=INPUT$(1):IF ASC(R$) <> 13 GOTO 510

520 CONSOLE ,,,1:COLOR 4,6:
    FOR I=0 TO 30:PRINT:NEXT:CONSOLE ,,,0:CLS
```

*** 500

Affichage en caractères de teille normale, Rouge (1) sur fond Bleu (4) de la chaîne "Tapez ".

On notera que la chaîne "Tapez" est suivie d'un espace, ce qui permet, l'ordre PRINT étant suivi d'un point-virgule (;), d'afficher la chaîne "ENTREE" en séquence, sans saut à la ligne. La chaîne "ENTREE" est affichée en Noir (0) sur fond Jaune (3).

*** 510

Mise en attente du programme jusqu'à ce que l'opérateur tape sur la touche "ENTREE" et sur elle seule. Toute autre touche (de numéro de code ASCII différent de 13) provoque un branchement au début de l'instruction pour une nouvelle lecture.

*** 520

a/ Nous trouvons ici l'un des nombreux cas d'utilisation de l'instruction CONSOLE -(M.Ref 35)-.

Le quatrième paramètre est mis égal à 1, ce qui met l'écran en mode "défilement doux". On notera que pour bien préciser que c'est le quatrième paramètre qui est mis à 1, l'instruction CONSOLE est suivie de trois virgules (,) puis du "1".

b/ Remet les couleurs caractères au Bleu (4) et la couleur fond au Cyan (6).

Cette instruction est indispensable sous peine de voir apparaitre un fond Jaune (4), imposé par le dernier COLOR en 500, après disparition de la mire.
On aurait pu penser à mettre ici une instruction SCREEN 4,6,6. Il faut prendre garde à ce que l'effet de SCREEN est immédiat et provoque le passage de tous les caractères au Bleu et de tous les fonds au Cyan. Par contre l'instruction COLOR n'a d'effet que sur les affichages de caractères suivants.

c/ d/ et e/ Boucle produisant 30 sauts de ligne, provoquant la montée en "défilement doux" de la mire.

f/ Remet l'écran en défilement normal.

Cette instruction est indispensable, sous peine de ne pouvoir lancer le programme suivant qu'avec le "défilement doux".

g/ Efface l'écran.

En définitive le programme est le suivant :

```
10 CLS:SCREEN 4,6,6:LOCATE 0,0,0

20 X0=160:Y0=96

30 BOX (0,0)-(319,192),1

40 BOXF (X0-80,Y0-80)-(X0+80,Y0+80),6

50 FOR Y=16 TO 176 STEP 16:LINE (0,Y)-(319,Y):NEXT

60 FOR X=16 TO 64 STEP 16:LINE (X,1)-(X,191):NEXT
```

```
70 FOR X=80 TO 224 STEP 16:
   LINE (X,1)-(X,Y0-80):
   LINE (X,Y0-79)-(X,Y0+80),-5:
   LINE (X,Y0+81)-(X,191):NEXT

80 FOR X=240 TO 304 STEP 16:
   LINE (X,1)-(X,191):NEXT

100 R=78:P=57.3

110 FOR T=1 TO 45 STEP 2

120  TT=T/P:C=COS(TT):S=SIN(TT)

130  X1=X0+R*C:Y1=Y0-R*S:X2=X0-R*C:Y2=Y0+R*S

140  X3=X0+R*S:Y3=Y0-R*C:X4=X0-R*S:Y4=Y0+R*C

150  BOXF (X1,Y1)-(X2,Y2),-5:
     BOXF (X3,Y3)-(X4,Y4),-5

160 NEXT

200 ATTRB 1,1:COLOR 7,4:LOCATE 16,4:PRINT "MIRE"

210 LOCATE 16,6:PRINT "TO 7"

220 A$=CHR$(22):COLOR 5,4:LOCATE 16,20:
    PRINT A$;",";A$;"-";A$;".";A$;"/":ATTRB 0,0

300 BOXF (112,64)-(215,127),4:
    BOX (103,63)-(216,128),1

310 FOR J=0 TO 1

320  FOR I=0 TO 3

330  X=113+24*I:C=15+3*I:U=I+4*J:CC=1+U:CCN=-CC:
     IF CC=8 THEN CC=0

340  BOXF (X,70+30*J)-(X+22,92+30*J),CCN

350  LOCATE C,10+4*J:COLOR CC,U:PRINT CHR$(48+U)

360  NEXT I

370 NEXT J
```

```
380 BOX (112,100)-(135,122),7

400 ATTRB 1,1:COLOR 4,6

410 FOR I=0 TO 19:LOCATE 2*I,1:
    PRINT CHR$(65+I):NEXT

420 FOR I=0 TO 5:LOCATE 2*I+14,23:
    PRINT CHR$(85+I):NEXT

430 FOR I=0 TO 4:LOCATE 0,2*I+9:
    PRINT CHR$(48+I):NEXT

440 FOR I=5 TO 9:LOCATE 38,2*(I-5)+9:
    PRINT CHR$(48+I):NEXT

450 BOX (0,0)-(319,192),4:LINE (0,176)-(319,176)

460 LINE (0,16)-(319,16):LINE (0,176)-(319,176)

470 FOR X=16 TO 304 STEP 16:
    LINE (X,0)-(X,16):LINE (X,176)-(X,192):NEXT

480 FOR Y=64 TO 128 STEP 16:
    LINE (0,Y)-(16,Y):LINE (304,Y)-(319,Y):NEXT

490 LINE (304,0)-(304,192)

500 ATTRB 0,0:LOCATE 14,17:COLOR 1,4:
    PRINT "Tapez ";:COLOR 0,3:PRINT "ENTREE"

510 R$=INPUT$(1):IF ASC(R$) <> 13 GOTO 510

520 CONSOLE ,,,1:COLOR 4,6:
    FOR I=0 TO 30:PRINT:NEXT:CONSOLE ,,,0:CLS
```

5

Conception de caractères graphiques

Nous savons qu'il est possible de définir puis d'utiliser des caractères graphiques venant compléter les caractères typographiques disponibles. Il est ainsi loisible de créer des alphabets grec, hébreu ou russe ou de créer des symboles mathématiques permettant une meilleure présentation sur écran.

La "fabrication" d'un caractère est précisée dans le manuel de référence. -(M. Ref 100)-

Résumons rapidement la méthode.

On dessine le graphique dans une matrice 8 lignes, chaque ligne étant composée de 8 carrés, chaque carré représentant un point de l'écran. On marque chaque carré faisant partie du graphisme. Pour chaque ligne on calcule la valeur correspondant au nombre binaire obtenu en prenant 1 pour un carré marqué et 0 pour un carré non marqué. On obtient ainsi 8 valeurs binaires (que l'on peut transcrire en valeurs décimales), que l'on classe dans l'ordre du haut vers le bas du graphisme.

On définit ainsi le caractère numéro N (N peut varier de 0 à 127) par l'instruction:

```
DEFGR$(N) = (liste)
```

où la liste est constituée des 8 valeurs déterminées précédemment en les séparant par des virgules.

Il est préférable de donner des valeurs décimales car cela prend moins de place et comporte moins de risques d'erreur de frappe: il est plus facile de taper 179 que &B10110011.

Une fois le caractère N ainsi défini en début de programme on peut l'utiliser par une instruction:

```
PRINT GR$(N)
```

Attention! Il ne faut pas oublier de réserver de la place en mémoire par une instruction CLEAR placée en début de programme -(M. Ref 32)-.

Il est recommandé de placer l'instruction CLEAR en tout début de programme car elle provoque en outre l'effacement de toutes les variables précédemment définies ou calculées (nous y reviendrons lors des explications détaillées des instructions).

Si par exemple nous voulons créer les vingt quatre lettres de l'alphabet grec, nous placerons l'instruction CLEAR ,,24 (notez les deux virgules) suivie des vingt quatre instructions DEFGR$(I), avec I variant de 0 à 23.

Nous trouverons un exemple d'utilisation de ces instructions dans le programme CLAVIER.

La difficulté dans la création de graphismes vient de ce qu'il n'est pas évident de prévoir quelle "allure" aura réellement le caractère, une fois présenté à l'écran. Le changement d'échelle entre la matrice tracée sur une feuille de papier et le carré de 8 x 8 points d'écran est trop important pour évaluer correctement l'effet produit.

Nous nous proposons donc d'écrire un programme qui nous permette

- de placer, à l'aide du crayon optique, nos petits carrés sur une matrice 8 sur 8 assez grande affichée sur l'écran,
- de faire le calcul des huit nombres caractéristiques,
- de tracer le caractère sur l'écran (à l'échelle normale dans un carré 8 x 8 points d'écran et à l'échelle double dans un carré de 16 x 16 points)
- et enfin de mettre en mémoire les huit nombres dès que l'on obtient un résultat satisfaisant, avant de passer à la conception du caractère suivant.

En fin de création de tous les caractères, on réaffichera les séries de valeurs caractéristiques pour permettre de les noter (une autre version du programme permettra de les relever directement sur l'imprimante).

L'écran étant composé de 25 lignes et de 40 colonnes, nous tracerons une matrice carrée de 8 cases sur 8 cases, chaque case occupant 3 lignes et 3 colonnes, de telle sorte que nous représenterons une case "occupée" en y plaçant au centre un carré "plein" de 8 points sur 8 points (ce carré plein est un caractère graphique dont les huit nombres caractéristiques sont tous égaux à 255).

On effacera une case occupée en y plaçant un carré "vide" de 8 points sur 8 points (ce carré vide est un caractère graphique dont les huit nombres caractéristiques sont tous égaux à 0).

Le carré plein, GR$(0), est alors défini par l'instruction:

```
DEFGR$(0) = 255, 255, 255, 255, 255, 255, 255, 255
```

puisque chacune des huit lignes du caractère est représentée par le nombre binaire 11111111 soit 255.

Le carré vide (ou caractère d'effacement), GR$(1), est défini par l'instruction:

```
DEFGR$(1) = 0, 0, 0, 0, 0, 0, 0, 0
```

puisque chacune des huit lignes du caractère est représentée par le nombre binaire 00000000 soit 0.

Nous matérialiserons la matrice par une grille de 9 lignes horizontales et de 9 lignes verticales, chaque ligne étant éloignée de la ligne précédente par 3 x 8 = 24 points d'écran.

La grille se présente alors comme indiqué sur la figure (Page 67), avec le coin haut-gauche placé en X = 56 et Y = 0.

Le marquage du carré, en colonne 2 ligne 3 de la matrice, se fera donc par une instruction LOCATE 11,7 suivie d'une instruction PRINT GR$(0).

D'une façon générale le marquage du carré en colonne C, ligne L de la matrice se fera par une instruction LOCATE CC,LL avec CC = 3 x C + 5 et LL = 3 x L - 2.

Pour noter que le carré en colonne C, ligne L de la matrice est marqué, la variable A(C,L) prend la valeur 1, sinon elle prend la valeur 0. Chaque fois que l'on bâtira un nouveau caractère il sera nécessaire d'effacer tous les marquages sur la grille et de remettre à 0 tous les A(C,L).

Pour piloter le programme, nous placerons 3 "boites" sur la gauche de l'écran, marquées par les trois lettres G,S et F.

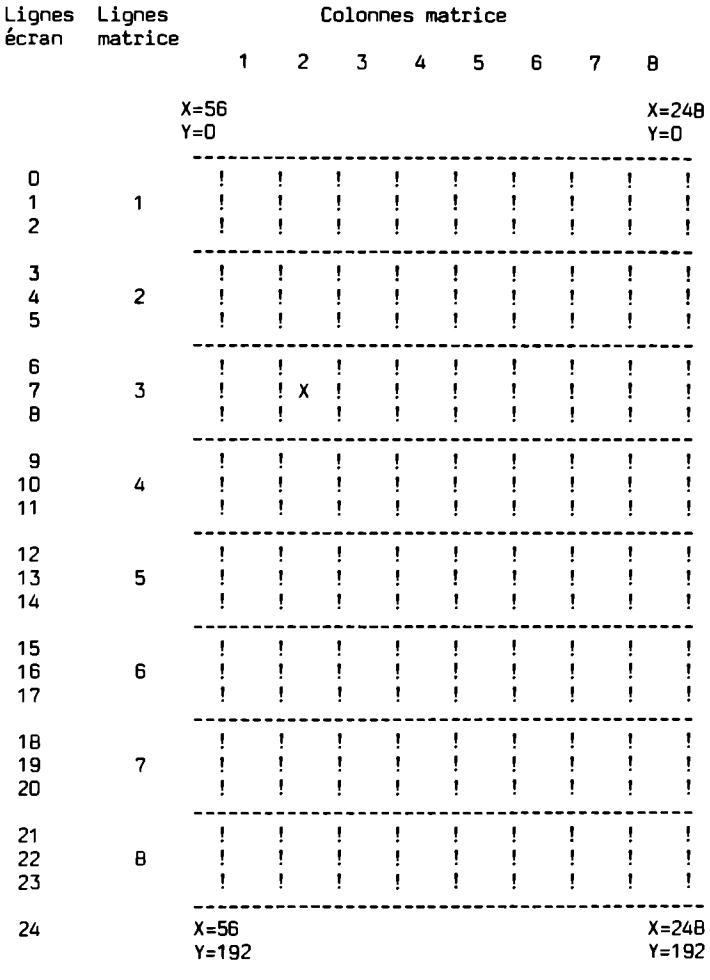
La désignation par le crayon optique de la boite G (pour graphique) provoquera le calcul des huit valeurs caractéristiques du caractère, leur affichage à droite de l'écran en face de chaque ligne de la matrice et l'affichage du caractère en bas et à gauche de l'écran en taille normale et en taille double.
Les huit valeurs caractéristiques du caractère de numéro NG sont stockées dans la variable Z(L,NG) où L varie de 1 à 8 et NG de 0 pour le premier caractère à une valeur maximale de 127.

La désignation au crayon optique de la case S (pour suivant) effacera la matrice, remettra à zéro les A(C,L) et incrémentera de une unité l'indice repère NG du caractère graphique (les huit valeurs de Z(L,NG) pour le caractère que l'on vient de bâtir sont alors définitivement stockées).

Enfin la désignation au crayon optique de la case F (pour fin) provoquera la récapitulation des résultats (ou leur impression sur l'imprimante).

Ces principes étant adoptés, il nous reste à établir le programme les traduisant en instructions utilisables par notre TO7.

On prendra garde à ne pas croire que les dispositions adoptées pour la présentation de la matrice, des "boites" de commandes G, S et F et pour la présentation des résultats ont été déterminées ex abrupto. En réalité un certain nombre d'essais ont été nécessaires pour parvenir à un résultat satisfaisant sur les plans esthétique et pratique. D'autres présentations sont possibles. Chacun pourra les modifier à sa convenance à condition de modifier la suite du programme en conséquence (cela constituera par ailleurs un excellent exercice!).



- OESSIN OE LA MATRICE -

Nous allons tout d'abord étudier le premier bloc du programme permettant de bâtir la grille.

```
10 CLEAR ,,3

20 DEFINT A-Z:DIM A(8, 8),Z(8, 127):NG= -1

30 NG=NG+1:CLS:SCREEN 1,6,6:LOCATE 32,0:
   PRINT USING "GR$###";NG:COLOR 4

40 DEFGR$(1)= 0,0,0,0,0,0,0,0:
   DEFGR$(0)= 255,255,255,255,255,255,255,255

50 FOR I= 0 TO 8

60 X=24*I+56

70 LINE (X,0)-(X,192)

80 NEXT

90 FOR I= 0 TO 8

100 Y=24*I

110 LINE (56,Y)-(248,Y)

120 NEXT

130 ATTRIB 1,1:BOX (0,0)-(32,32):
    LOCATE 1,2,0:PRINT "G"

140 BOX (0,40)-(32,72):LOCATE 1,7:PRINT "S"

150 BOX (0,80)-(32,112):
    LOCATE 1,12:PRINT "F":ATTRB 0,0

160 FOR I=1 TO 8:FOR J=1 TO 8:A(I,J)=0:NEXT J,I

170 BOX (0,0)-(319,199)

180 BEEP:FOR I=0 TO 100:NEXT:BEEP
```

*** 10
L'instruction CLEAR,,3 -(M.Ref 32)-

-d'une part efface toutes les variables et c'est pourquoi nous la plaçons en début de programme de façon à ne pas provoquer de catastrophes par effacement de variables préalablement définies ou calculées.

-d'autre part réserve 3 places pour les caractères graphiques (soit une place pour GR$(0) qui est le carré 8 x 8 plein, une place pour GR$(1) qui est le carré 8 x 8 vide et une place pour le caractère GR$(2) qui sera défini par l'opérateur en cours d'éxécution).

On n'oubliera pas les deux virgules avant le "3".

Toute instruction CLEAR fixe trois paramètres par CLEAR I,J,K.

Dans notre exemple nous n'avons fixé que la valeur de K (soit 3) qui est le nombre de caractères graphiques. C'est pour préciser que la valeur 3 choisie était le nombre de caractères, que nous l'avons fait précéder des deux virgules.

I est le nombre de places mémoires réservées pour les manipulations de chaines de caractères.
Nous ne manipulons pas de chaines de caractères, cette réservation n'est donc pas utile (la valeur prise par défaut pour I est 300).

La valeur de J est l'adresse maximale que peut utiliser l'interpréteur BASIC.
Nous n'utilisons aucun sous programme en assembleur ni fichiers binaires. Aussi n'est-il pas nécessaire de fixer la valeur de J. Dans ce cas l'interpréteur BASIC utilise toute la place mémoire disponible.

*** 20
a/ L'instruction DEFINT A-Z -(M.Ref 40)- conduit à n'utiliser que des variables entières d'où un gain de place mémoire pour les tableaux A(C,L) et Z(L,NG).

N'oublions pas qu'une variable réelle simple précision occupe 4 octets alors qu'une variable entière n'occupe que 2 octets. Le gain n'est pas considérable, sauf lorsque nous utilisons des tableaux.

Les variables A(C,L) et Z(L,NG) (voir paragraphe (b) suivant) nécessitent respectivement la réservation de 9 x 9 = 81 et 9 x 128 = 1152 places mémoires soit en tout 1233 places qui représentent l'occupation de 2466 octets en variables entières et de 4932 octets en variables réelles.
On voit tout de suite le gain de mémoire (2466 octets) à n'utiliser que des variables entières.

b/ L'instruction DIM A(8,8),Z(8,127) -(M.Ref 46)- réserve (8 + 1) x (8 + 1) = 81 places mémoires pour A(C,L); C et L peuvent ainsi varier de 0 à 8; nous n'utiliserons pas la valeur 0 car il est plus commode de numéroter les lignes de 1 à 8 plutôt que de 0 à 7.

Nous remarquerons que lorsque l'on manque de place mémoire il y a intérêt à utiliser les valeurs 0 des indices (surtout sur les tableaux jouant sur plus de deux indices).

L'instruction DIM réserve en outre (8 + 1) x (127 + 1) = 1152 mémoires pour la variable Z(L,NG), L pouvant varier de 0 à 8 et NG de 0 à 127 (on constate ici que nous gâchons 128 places mémoires inutilisées pour la valeur 0 de L).

c/ Nous initialisons le compteur NG des caractères à la valeur -1.

Cette initialisation est nécessaire car à l'instruction 30 suivante (qui sera l'instruction de retour une fois défini le caractère, voir étiquette 1010), NG est incrémenté de une unité. Ainsi le premier caractère construit aura le numéro de compteur 0 et ses valeurs caractéristiques stockées dans les 8 variables Z(L,0) avec L variant de 1 à 8.

*** 30
a/ Incrémentation du compteur NG.

b/ L'instruction CLS -(M.Ref 34)- efface l'écran

c/ L'instruction SCREEN 1,6,6,-(M.Ref 193)- impose des caractères Rouge (1) sur écran Cyan (Bleu clair) (6) avec un cadre extérieur Cyan (6). Cette instruction est nécessaire pour préciser la couleur de l'écran et du cadre extérieur. Nous modifierons ultérieurement la couleur des caractères par l'instruction COLOR (voir le paragraphe (f) de cette instruction).

d/ L'instruction LOCATE 32,0 -(M.Ref 72)- place le curseur en colonne 32, ligne 0 (donc en haut et à droite de l'écran).

e/ L'instruction PRINT USING avec une image composée de la chaine GR$ suivie de trois caractères dièses -(M.Ref 90)- affiche à l'écran le message GR$ avec le G en colonne 32, ligne 0 (imposé par le LOCATE précédent) et inscrit la valeur numérique de NG (3 chiffres au plus) avec le chiffre des unités en colonne 37. Les 3 places pour les 3 chiffres de NG sont précisées par les 3 caractères dièses de l'image. Ainsi en début de programme GR$ 0 sera affiché en Rouge sur fond d'écran 8leu.

f/ L'instruction COLOR 4 -(M.Ref 189)- change en 8leu la couleur des caractères qui seront affichés après cette instruction.

On notera que les deuxième et troisième paramètres qui figurent normalement dans une instruction COLOR ne sont pas indiqués.

Dans ce cas la couleur du fond de caractère est prise égale par défaut à la dernière couleur de fond de caractère préalablement définie par une précédente instruction COLOR ou, ce qui est notre cas, à la couleur de l'écran imposée par l'instruction SCREEN.

On prendra garde à ne pas confondre la couleur du fond de caractère et la couleur de l'écran: le fond de caractère est le rectangle de 8 x 8 points (ou 16 x 16 pour les grands caractères) dans lequel est inscrit le caractère. C'est la couleur de ce rectangle qui est fixée par le deuxième paramètre de COLOR indépendemment de la couleur générale de l'écran (voir Annexe I).

*** 40

Les deux instructions DEFGR$(1) et DEFGR$(0) -(M.Ref 190)- définissent GR$(1), le carré vide 8 x 8, et GR$(0), le carré plein.

*** 50-80

Ces quatre instructions permettent de tracer 9 lignes verticales entre les points d'ordonnées Y=0 et Y=192 avec des abscisses X variant de 24 en 24 depuis X=56 jusqu'à X= 192+56 = 248

On aurait pu obtenir le même résultat par la suite d'instructions suivantes:

```
50  FOR X=56 TO 248  STEP 24

60  LINE (X,0)-(X,192)

70  NEXT
```

En pratique il est plus commode de conserver les instructions précédentes car elles permettent de faire varier simplement la position de la grille verticale au moment de la mise au point du programme (recherche de la position de matrice la plus commode et la plus esthétique).

Pour la boucle FOR NEXT -(M.Ref 54)-, on notera que l'instruction 80 se réduit à NEXT, et non pas NEXT I, ce qui est licite, puisqu'il n'y a qu'une seule boucle en service.

Pour l'instruction LINE (70) -(M.Ref 187)- on n'a pas précisé la valeur de la couleur (4 pour le Bleu) puisque la couleur des caractères a déjà été précisée par l'instruction COLOR 4 en 30 (f).

*** 90-120

Ces quatre instructions analogues à 50-80 permettent de tracer 9 lignes horizontales entre les points d'abscisses X = 56 et X= 248 avec des ordonnées Y variant de 24 en 24 depuis Y= 0 jusqu'à Y= 192.

Là encore on aurait pu simplifier avec:

```
90 FOR Y=0 TO 192 STEP 24

100 LINE (56,Y)-(248,Y)

110 NEXT
```

*** 130

a/ L'instruction ATTR8 1,1 -(M.Ref 26)- fixe la taille des caractères qui s'inscriront dans un carré de 16 x 16 points d'écran. Cela permettra de tracer les caractères G,S et F destinés à la commande, avec une taille qui les assimile à une "touche" sur l'écran.

b/ L'instruction BOX (0,0)-(32,32) -(M.Ref 187)- dessine un cadre carré de 32 points de coté avec le coin supérieur gauche au point 0,0 (coin supérieur gauche de l'écran). La couleur du cadre n'a pas été précisée (comme pour l'instruction LINE en 70 et 110 qui a la même syntaxe) car nous la désirons Bleu comme pour les caractères.

c/ L'instruction LOCATE 1,2,0 -(M.Ref 72)- place le curseur en colonne 1, ligne 2 de l'écran. Le dernier 0 impose au curseur de ne plus être visible. La position du curseur n'est en effet plus utile puisque nous n'interviendrons pas avec le clavier mais avec le crayon optique).

d/ L'instruction PRINT "G" -(M.Ref 87)- affiche le caractère G avec le carré bas-gauche des quatre carrés 8 x 8 points constituant le carré 16 x 16 dans lequel s'inscrit le caractère en colonne 1, ligne 2 (imposé par le LOCATE 1,2,0 précédent). Ainsi le caractère G se trouve centré dans le cadre 32 x 32 points dessiné par l'instruction 130 (b).

*** 140-150
Instructions analogues à 130 permettant de tracer les "touches" S et F au dessous de la touche G.

*** 160
Oouble boucle permettant de mettre à zéro les 64 valeurs du tableau A(I,J).

Cette instruction ne serait pas nécessaire si l'on ne bâtissait qu'un seul caractère puisque toutes les variables sont initialisées à zéro par l'instruction RUN.

Par contre pour dessiner un nouveau caractère il est nécessaire d'annuler préalablement les A(I,J).

On notera l'instruction NEXT J,I qui est équivalente aux deux instructions NEXT J:NEXT I.

On prendra garde à bien écrire J et I dans cet ordre pour respecter l'emboitement des deux boucles en J et I.

*** 170
Trace le cadre délimitant la partie utile de l'écran.

L'objectif en est esthétique plutôt que fonctionnel.

*** 180
L'instruction BEEP -(M.Ref 30)- produit un son court au haut-parleur.

La boucle de retardement FOR I=0 TO 100:NEXT impose un certain délai entre le premier Bip et le second. En son absence on obtient un Bip théoriquement deux fois plus long qu'un Bip unique mais qui, en pratique, ne s'en distingue pas.
Ce double Bip est destiné à signaler à l'opérateur que la mise en place de la matrice est terminée (la remise à zéro du tableau A(I,J) demande un délai court mais perceptible).

Si l'on désire un signal plus intense on peut remplacer l'instruction 180 par la suivante:

```
180 FOR I=0 TO 50:BEEP:NEXT
```

qui produit un signal continu dont la durée est imposée par la borne supérieure de la boucle en I.

Le programmeur choisira par essais successifs la valeur qui lui semble la plus adaptée.

* * * * * *

L'ensemble des instructions 30 à 180 permet donc de mettre en place la matrice et les touches de commande G, S et F.

C'est maintenant l'occasion pour le programmeur d'essayer le programme.

Cela lui permet d'éliminer les erreurs de syntaxe dues en général à des fautes de frappe (parenthèses remplacées par un 8 ou un 9 par oubli de la touche majuscule, virgules oubliées, point-virgule et deux-points intervertis, etc..).
Cela lui permet également d'apprécier l'esthétique et la commodité de la présentation. Il peut préférer déplacer la matrice à droite ou à gauche, placer les touches G, S et F ailleurs etc. Cela n'empêchera pas des modifications après essais d'exploitation (on peut alors estimer plus commode une autre disposition), mais cela permet de poursuivre la mise au point sur des bases plus solides.

Pour bien identifier ce bloc d'instructions ayant une fonction de dessin, nous allons poursuivre la numérotation des étiquettes en sautant à l'étiquette 200 (souci de clarté de présentation du programme, ce qui n'est pas inutile lorsque l'on est amené à effectuer des améliorations ultérieures).

```
200 INPUTPEN X,Y

210 FOR J=1 TO 10:IF PTRIG GOTO 210 ELSE NEXT J

220 IF X<0 OR Y<0 GOTO 200

230 IF X<33 GOTO 1000

240 C=INT((X-56)/24)+1:L=INT(Y/24)+1:
    CC=3*C+5:LL=3*L-2

250 IF C<1 OR C>8 OR L<1 OR L>8 GOTO 200

260 LOCATE CC,LL:PRINT GR$(A(C,L))

270 A(C,L)=-(A(C,L)=0)

280 GOTO 200
```

*** 200

L'instruction INPUTPEN X,Y -(M.Ref 63)- permet de recueillir les coordonnées X et Y du point de l'écran sur lequel le crayon optique a été appuyé (l'information en X et Y n'est prise en compte que si l'interrupteur du crayon optique est fermé, c'est-à-dire si le crayon a été appuyé sur l'écran).

Si l'interrupteur a été fermé par appui sur une surface quelconque en dehors de la zone de travail de l'écran (c'est-à-dire à l'extérieur du cadre défini par l'instruction 170), X et Y prennent la valeur -1. Cette remarque nous sera utile pour expliquer l'instruction 220.

*** 210

Supposons que le crayon optique soit manipulé maladroitement et que l'opérateur, tout en plaçant le crayon devant la zone appropriée, appuie rapidement deux fois sur l'interrupteur.

Or il se trouve que les instructions 220 à 280, destinées en général à afficher un carré plein (ou à l'effacer) dans la case de la matrice pointée par le crayon, sont exécutées très rapidement avec retour à l'instruction 200 (par l'instruction 280) pour une nouvelle lecture de position. Tout porte à croire que la deuxième lecture de X et de Y effectuée en 200 à la deuxième fermeture maladroite de l'interrupteur produira l'effet contraire de la première lecture et l'annulera.

Il est donc nécessaire d'établir une courte temporisation pendant laquelle toute action sur l'interrupteur sera sans effet.

C'est l'objet de l'instruction 210.

Une boucle en J est déclenchée par :

```
FOR J=1 TO 10
```

C'est elle qui assure la temporisation, le temps pour la machine de compter de 1 à 10. Mais le compteur est remis à 1 si au cours du comptage l'interrupteur est fermé, c'est-à-dire si le crayon est à nouveau appuyé.

C'est l'objet de la deuxième instruction de 210.

```
IF PTRIG GOTO 210 ELSE NEXT J
```

(voir -(M.Ref 135)- pour PTRIG), qui renvoie à la première instruction de 210 (c'est-à-dire à la mise à 1 du compteur J) si PTRIG est égal à Vrai (-1), donc si l'interrupteur est fermé, sinon poursuit la boucle en J.

La valeur 10 choisie pour la fin du comptage de temporisation a été déterminée expérimentalement. Elle ne retarde pas trop l'affichage et évite l'effet de rebond du crayon ce que ne permettrait pas une valeur plus faible. Mais rien n'empêche de choisir une valeur différente mieux adaptée à chaque opérateur.

*** 220

Renvoie à la lecture de X et Y (instruction 210) si l'interrupteur a été fermé en appuyant le crayon optique en dehors de la zone de travail écran (l'instruction INPUTPEN X,Y renvoie X= -1 et Y= -1 dans ce cas -(M.Ref 63)-).

*** 230

Si la valeur de X relevée par l'instruction 200 est inférieure à 33, c'est que le crayon optique a été appuyé sur la gauche de l'écran dans la zone des "touches" G, S ou F. Dans ces conditions le programme saute à l'instruction 1000 où l'on identifiera la touche choisie par un test sur la valeur de Y.

Remarque.

Les valeurs de X transmises par le crayon optique ne sont pas "continues".

Elles ne peuvent être que 3, 11, 19, 27,.. (avec un pas de 8). La valeur 35, par exemple, est transmise si le crayon optique a été appuyé en face des points d'abscisses 32 à 39.

La valeur 33 de l'instruction 230 aurait donc pu être remplacée par toute valeur comprise entre 28 et 34, puisque la valeur 27 est obtenue pour tous les points d'abscisses allant de 24 à 31 (acceptable) et la valeur 35 pour tous les points d'abscisses allant de 32 à 39 (inacceptable).

*** 240

a/ Calcul du numéro de colonne C de la case de la matrice pointée par le crayon optique.

La fonction INT(U) -(M.Ref 126)- calcule le plus grand entier inférieur ou égal à l'argument U.

Ainsi le crayon ayant été placé sur la deuxième case, X peut prendre les valeurs 83, 91 et 99 et (X-56)/24 prend les valeurs 1.125, 1.458, et 1.792 dont la partie entière est 1 ce qui donne C = 2.

b/ Calcul du numéro de la ligne L de la case de la matrice pointée par le crayon optique.

La formule est analogue à la précédente (à noter que Y, contrairement à X, prend toutes les valeurs entières comprises entre 0 et 199).

c/ et d/ Calcul des coordonnées CC et LL de la case écran (CC varie de 0 à 39 et LL de 0 à 24) au centre de la case matrice pointée (nous avons vu le calcul à l'occasion de la préparation du programme).

*** 250
Branchement en 200 pour un autre pointé, dans le cas où le pointé précédent a été fait en dehors de la grille.

*** 260
a/ Positionne le curseur (invisible depuis l'instruction 130) en colonne CC et en ligne écran LL.

b/ Affiche le caractère graphique GR$(0) ou GR$(1) suivant que A(C,L) est égal à 0 ou à 1. Or A(C,L) = 0 signifie que la case C,L de la matrice est "vide"; dans ce cas c'est GR$(0) (carré plein) qui s'affiche. Si au contraire A(C,L) = 1 la case C,L de la matrice est "pleine" et l'affichage de GR$(1) (carré vide) vient effacer le carré plein qui y figure.

*** 270
L'expression (A(C,L)=0) est une variable logique -(M.Ref 20)- qui vaut -1 (vrai) si A(C,L) est égal à 0 et qui vaut 0 (faux) si A(C,L) est égal à 1.

Ainsi l'instruction 270 change A(C,L) de 0 en 1 ou de 1 en 0.

En conséquence les deux instructions 260 et 270 permettent de changer l'état, "plein" ou "vide", de la case pointée en l'état contraire à la fois sur l'écran et en mémoire.

Remarque.

L'instruction 270 pourrait également s'écrire

A(C,L) = - NOT (-A(C,L)) -(M.Ref 21)-

puisque si A(C,L) = 0 alors NOT (-A(C,L)) = -1
et si A(C,L) = 1 alors NOT (-A(C,L)) = 0

*** 280
Branche en 200 pour un nouveau pointé.

* * * * * *

L'ensemble des instructions 200 à 280 forme un bloc dont la fonction est de changer l'état des cases de la matrice (du moins tant que le crayon optique n'est pas pointé sur l'une des cases de commande G, S ou F). Il est donc possible, une fois tapé cet ensemble, de l'essayer. Toutefois pour vérifier que l'instruction 230 branche effectivement en 1000 si l'on pointe l'une des trois cases G, S ou F, on place l'instruction provisoire:

```
1000 CLS :PRINT "1000"
```

Pour finir les vérifications du bloc de changemant d'état, on pointera donc sur l'une des cases de commande. On doit alors observer l'effacement de l'écran (CLS) et l'affichage de "1000" dans le coin haut gauche de l'écran.

* * * * * *

Lorsque l'opérateur a fini de disposer ses points sur la grille, il commande la visualisation du caractère en appuyant sur la "touche" G (avec le crayon optique), ce qui a pour effet de brancher le programme en 1000, où un test sur la valeur de Y détectée par le crayon permet de brancher sur la partie de programme chargée de la visualisation.

C'est ce bloc de programme, commençant en 500 que nous allons étudier maintenant (nous verrons plus tard le bloc des instructions 1000).

```
500 COLOR 1:FOR L=1 TO 8

510 Z=0

520 FOR C=1 TO 8

530 Z=2*Z+A(C,L)

540 NEXT C :Z(L,NG)=Z

550 LOCATE 36,3*L-2:PRINT USING "###";Z:BEEP

560 NEXT L
```

```
570 DEFGR$(2)= Z(1,NG),Z(2,NG),Z(3,NG),Z(4,NG),
               Z(5,NG),Z(6,NG),Z(7,NG),Z(8,NG)

580 BOX (0,160)-(32,192)

590 ATTRB 1,1:LOCATE 1,22:PRINT GR$(2):ATTRIB 0,0:
    LOCATE 5,22:PRINT GR$(2)

600 COLOR 4

610 GOTO 200
```

*** 500

Cette instruction ne peut être atteinte que par un branchement imposé en 1000.

Le bloc 500 à 610 a pour fonction de calculer les huit valeurs caractérisant les huit lignes du caractère bâti et de dessiner ce caractère sur l'écran pour juger l'effet produit. Il est donc exécuté chaque fois que la case de commande "G" a été pointée (voir instructions 1000 et 1010).

a/ Change la couleur du caractère (Rouge 1), sans changer la couleur du fond qui reste Cyan.

b/ Déclanche une boucle sur L (L variant de 1 à 8). Il s'agit de calculer les nombres caractérisant les 8 lignes de points composant le caractère.

*** 510

La variable Z doit pour chaque valeur de L prendre la valeur caractéristique de la ligne. Elle est mise à zéro en début de calcul.

*** 520 - 540

Boucle en C (C variant de 1 à 8) destinée à explorer les A(C,L) et à calculer la valeur caractéristique Z pour la ligne L

Si a,b,c,d,e,f,g,h sont les valeurs de A(C,L) pour C variant de 1 à 8 (ligne L)(avec a,b,c,d,e,f,g,h,= 0 ou 1), alors

$$Z=(((((((0+a) \times 2+b) \times 2+c) \times 2+d) \times 2+e) \times 2+f) \times 2+g) \times 2+h$$

Ces opérations de multiplication par 2 et d'addition de la valeur suivante de A(C,L) sont éxécutées dans la boucle.

Enfin la dernière instruction de 540 place le valeur Z trouvée pour la ligne L dans la variable Z(L,NG) où NG est le numéro du caractère bâti.

Remarque.

On notera que les valeurs Z et Z(L,NG) sont deux variables différentes. L'une est une variable simple, l'autre est un tableau.

*** 550

a) Positionne le curseur en colonne 36 sur la ligne écran au milieu de la ligne L de la matrice (d'où 3 x L - 2 qui pour L = 2 par exemple donne bien la ligne d'écren 4).

b) Affiche les trois chiffres de Z (ou un espace et deux chiffres ou deux espaces et un chiffre suivant que Z est compris entre 10 et 100 ou inférieur à 10) avec les unités en colonne 38 (le premier caractère dièse de l'image , représentant le premier chiffre ou le premier blanc, étant positionné en colonne 36 par l'instruction LOCATE).

c) Déclenche un Bip sonore. Cette instruction n'est manifestement pas nécessaire, c'est pour l'esthétique (?).

*** 560

Clôture la boucle en L. On notera qu'il est impératif de préciser le nom de la variable puisqu'il y a deux boucles imbriquées.

*** 570

Les huit valeurs Z(L,NG) (L variant de 1 à 8) ayant été calculées, l'instruction définit le caractère GR$(2) qui est le nouveau caractère qui vient d'être conçu de façon à le présenter à l'écran. Ainsi GR$(2) prendra toutes les formes conçues successivement.

On notera que l'instruction DEFGR$ peut comporter des variables au lieu de valeurs numériques.

*** 580

Définit un cadre carré 32 x 32 de la couleur choisie en 500 pour les caractères (soit le Rouge 1), aucun numéro de couleur ne figurant à droite de l'instruction-(M.Ref 187)-

*** 590
a/ Fixe la taille des caractères (taille 16 x 16 points).

b/ Positionne le curseur en colonne écran 1 et ligne écran 22, de telle sorte que le caractère taille double se place au centre du cadre 32 x 32 défini en 580.

c/ Affiche le nouveau caractère conçu GR$(2).

d/ Fixe la taille des caractères (taille 8 x 8).

e/ et f/ Affiche GR$(2) taille normale en colonne 5 ligne 22.

On notera que les deux caractères sont affichés en Rouge (imposé par l'instruction COLOR de 500).

*** 600
Remet la couleur des caractères à la valeur 4 (Bleu).

*** 610
Branche en 200 pour un nouveau pointé.

Remarque.

Si l'on désire mettre en mémoire le caractère visualisé en 590, il suffit alors de pointer la case commande S (l'instruction 1010 branchera sur 30 où NG est incrémanté et où un nouveau calcul est préparé).

Si, au contraire, le caractère ne répond pas aux espoirs du concepteur, il suffit d'effectuer les modifications directement sur la matrice en cours (en général le déplacemant de quelques points suffit) et de repointer la case commande G pour voir l'effet produit.

Là encore les instructions 500 à 610 forment un bloc qu'il y a intérêt à essayer immédiatement.

* * * * * *

Regardons maintenant le bloc logique qui permat les branchements aux instructions répondant à la frappe "optique" des "touches" S et F.

```
1000 V=1-(Y(33)-2*(Y )39 AND Y(73)
       -3*(Y)79 AND Y(113)

1010 ON V GOTO 200,500,30,2000
```

*** 1000

Cette instruction est atteinte par branchement conditionnel depuis l'instruction 230, lorsque le pointé s'est fait avec une abscisse inférieure à 32, c'est-à-dire lorsque l'opérateur a pointé l'une des cases commandes G, S ou F.

Les variables (Y(33), (Y)39 AND Y(73) et (Y)79 AND Y(113) sont des variables logiques qui prennent la valeur 0 si l'expression entre parenthèses est fausse et -1 si elle est vraie -(M.Ref 20)-.

(Y (33) est égal à -1 si la case pointée est G (on a vérifié en 220 que Y ne peut être négatif).

(Y)39 AND Y(73) est égal à -1 si le case pointée est S.

Enfin (Y) 79 AND Y(113) est égal à -1 si la case pointée est F.

Au plus l'une de ces valeurs peut être égale à -1 (le pointé ne peut se faire dans deux cases à la fois).

En conséquence la variable V prend la valeur 1 si aucune case n'a été pointée (Le pointé a bien été effectué dans la "colonne" des cases commandes, puisque X est inférieur à 33 ce qui a provoqué le branchement en 1000, mais le crayon a été appuyé en dehors des "touches" commande). V prend la valeur 2 si la case G a été pointée, la valeur 3 si la case S a été pointée et enfin la valeur 4 si la case F a été pointée.

Cette valeur de V ainsi calculée va nous être utile pour brancher le programme suivant la case commande choisie.

*** 1010
L'instruction ON V GOTO -(M.Ref 80)- branche à la première étiquette (200) si V=1, à la deuxième étiquette (500) si V=2, à la troisième (30) si V=3, à la quatrième (1020) si V=4.

V=1 (erreur de pointé en dehors des cases commande) branche en 200 pour un nouveau pointé.

V=2 (commande G) branche en 500 (bloc 500-610 de calcul et d'affichage du caractère).

V=3 (commande S) branche en 30 (bloc 30-180, incrémentation du compteur de caractères NG, redessin de la matrice et mise à zéro des mémoires A(C,L).

V=4 (commande F) branche en 2000 (récapitulation des résultats).

Les instructions 1000-1010 forment un bloc dont il faut vérifier le fonctionnement (Il est commode de placer à ce stade une instruction provisoire en 2000, par exemple : 2000 CLS:PRINT "2000")

* * * * * *

Les instructions du bloc 2000-2100 ont pour objet d'afficher à l'écran la récapitulation des résultats, c'est-à-dire les listes des huit valeurs caractéristiques pour chaque caractère graphique.

```
2000 CLS:LOCATE 13,0:PRINT "RECAPITULATION":
     LOCATE 20,24:PRINT "TAPEZ ";:COLOR 7, 1:
     PRINT "ENTREE";:COLOR 4,6:CONSOLE 2,22

2010 N1=-8:S=0

2020 LOCATE 0,3:N1=N1+8:N2=N1+7:
     IF N2)=NG THEN N2=NG:S=1
```

```
2030 FOR I=N1 TO N2

2040 DEFGR$(2)=Z(1,I),Z(2,I),Z(3,I),Z(4,I),
                Z(5,I),Z(6,I),Z(7,I),Z(8,I)

2050 PRINT USING "GR$(###)=";I;:
     FOR J=1 TO 8:PRINT Z(J,I);",";:NEXT J

2060 PRINT "    ";:COLOR 1:PRINT GR$(2):COLOR 4

2070 NEXT I

2080 R$=INPUT$(1):R=ASC(R$):IF R<>13 GO TO 2080

2090 CLS:IF S=0 GO TO 2020

2100 CONSOLE 0,24:CLS
```

*** 2000

a/ Efface l'écran.

b/ et c/ Affiche "RECAPITULATION" au milieu de la ligne 0 de l'écran (la chaine "RECAPITULATION" a 14 caractères; il faut donc placer le premier caractère en colonne (40 - 14)/2 = 13).

d/ et e/ Affiche le message "TAPEZ" en ligne 24 avec le T en colonne 20.

On notera l'espace après le mot TAPEZ ainsi que le point-virgule (;). La chaine suivante sera alors affichée sur la même ligne avec un espace après TAPEZ.

f/ Change la couleur des caractères ainsi que la couleur du fond de caractère -(M.Ref 189)-.

La couleur des caractères passe au Noir (7) et la couleur du fond de caractère au Rouge (1).Ainsi le message suivant apparaitra en Noir sur un rectangle Rouge, le fond d'écran restant Cyan.

g/ Affiche le message ENTREE.

On notera là encore le point-virgule (;) après la chaine "ENTREE". Il interdit le saut à la ligne qui, effectué en ligne 24 provoquerait un déroulement intempestif de l'écran vers le haut avec disparition du message affiché en ligne 0.

h/ Retour à la couleur 8leu (4) des caractères avec un fond de caractère Cyan (6) identique au fond d'écran.

i) L'instruction CONSOLE -(M.Ref 35)- fixe les lignes d'écran ici 2 à 22 qui seront affectées par les instructions d'effacement ou de déroulement d'écran.

Ainsi les lignes 0 et 1, en haut de l'écran et les lignes 23 et 24 en bas ne seront pas effacées par l'instruction CLS en 2090 ce qui laisse permenents les messages affichés sur ces lignes.

*** 2010
a/ Initialise à -8 le compteur N1.

Les instructions 2030 à 2070 éxécutent une boucle permettant d'afficher les résultats par paquets de 8. Le compteur N1 est la valeur prise par l'indice I en début de boucle. Il est incrémenté de 8 pour préparer l'affichage d'un nouveau paquet de résultats. La première valeur de l'indice I étant 0, iL faut poser N1 = -8 pour obtenir N1 = 0 à l'instruction 2020 provoquant la première incrémentation de 8.

b/ Met à zéro l'indicateur S.

Cet indicateur sera mis à 1 lorsque le dernier résultat (pour I= NG) aura été affiché. Un test sur S permettra alors (instruction 2090) de se brancher sur l'instruction de fin de programme.

*** 2020
a/ Positionne le curseur en colonne 0, ligne 3.

b/ Incrémente de 8 le compteur N1 (indice de début d'affichage).

La première valeur affichée sera donc celle correspondant au caractère I = 0.

Après affichage des 8 x 8 valeurs correspondant aux caractères I = 0 à I = 7, le deuxième affichage débutera avec I = 8.

c/ Calcule la valeur N2 de l'indice de fin de paquet de huit résultats.

d/ Si la valeur N2 précédemment calculée dépasse ou égale la dernière valeur NG du compteur de caractère, alors N2 est posé égal à NG et l'indicateur S est mis égal à 1 (indication de fin d'effichage).

*** 2030-2070

Affichage des résultats par paquets de huit (sauf pour le dernier paquet qui est limité à NG).

* 2030
Ouverture de la boucle.

* 2040
Définit le caractère GR$(2) à partir des caractéristiques Z(1,I), Z(2,I), ..Z(8,I) du caractère numéro I. Ainsi GR$(2) prend successivement la forma de tous les caractères conçus.

* 2050
a/ Affiche la chaine "GR$( )" avec trois places disponibles entre les parenthèses pour y afficher la valeur I avec le chiffre des unités à la place de droite. Le point-virgule en fin d'instruction (c) évite le saut à la ligne.

b/ c/ et d/ Boucle en J permettant d'afficher les huit valeurs Z(1,I) à Z(8,I) séparées par des virgules. Là encore le point virgule en fin d'instruction (c) évite le saut à la ligne.

* 2060
a/ Affiche 3 espaces sans saut à la ligne.

b/ Passe la couleur des caractères au Rouge (1).

c/ Affiche le caractère GR$(2) qui a la forme du caractère graphique numéro I.

L'abscence de point-virgule en fin d'instruction provoque le saut à la ligne.

d/ Passe la couleur des caractères au Bleu (4).

* 2070
Fin de boucle en I.

*** 2080

a/ Attend la frappe au clavier d'un seul caractère et le place dans la variable R$ -(M.Ref 174)-.

b/ Place dans la variable R le numéro de code ASCII du caractère R$ -(M.Ref 111)-.

c/ Si la réponse attendue n'est pas le caractère correspondant à la touche ENTREE, c'est-à-dire le caractère de numéro de code ASCII 13, cette dernière instruction renvoie en 2080 donc à la première instruction (a). Si par contre la touche ENTREE a été frappée le programme saute à l'instruction suivante 2090 de poursuite d'affichage.

Ainsi l'instruction 2080 constitue un arrêt de déroulement du programme pour permettre la lecture des huit blocs de valeurs affichés. Le passage à l'affichage suivant se fait par frappe sur la touche ENTREE comme l'indique le message affiché à la ligne 24 de l'écran.

Si les deux instructions (b) et (c) de 2080 sont omises, l'instruction (a) provoque la même interruption de programme, mais la frappe de n'importe quelle touche supprime cette interruption.

Il est préférable d'imposer le redémarrage par la frappe d'une touche précise. Cela évite des fausses manoeuvres par appui intempestif sur le clavier.

*** 2090

a/ Efface l'écran dans la fenêtre ligne 2 à 22 imposée par l'instruction CONSOLE 2, 22 en 2000.

b/ Dans la mesure où l'affichage de tous les caractères n'est pas terminé (S=0) branche en 2020 pour le paquet suivant. Si S=1 le programme passe à l'instruction 2100.

*** 2100

a/ Annule l'effet de l'instruction CONSOLE 2,22 de 2000 en ne réservant aucune fenêtre.

b/ L'instruction CLS efface tout l'écran.

* * * * * *

Nous sommes maintenant en mesure d'essayer le programme au complet dont le listing est le suivant :

```
10 CLEAR ,,3

20 DEFINT A-Z:DIM A(8,8),Z(8,127):NG=-1

30 NG=NG+1:CLS:SCREEN 1,6,6:LOCATE 32,0:
     PRINT USING "GR$###";NG:COLOR 4

40 DEFGR$(1)= 0,0,0,0,0,0,0,0:
   DEFGR$(0)= 255,255,255,255,255,255,255,255

50 FOR I= 0 TO 8

60 X=24*I+56

70 LINE (X,0)-(X,192)

80 NEXT

90 FOR I= 0 TO 8

100 Y=24*I

110 LINE (56,Y)- (248,Y)

120 NEXT

130 ATTRIB 1,1:BOX (0,0)- (32,32):
    LOCATE 1,2,0:PRINT "G"

140 BOX (0,40)-(32,72):LOCATE 1,7:PRINT "S"

150 BOX (0,80)-(32,112):LOCATE 1,12:
    PRINT "F":ATTRB 0,0

160 FOR I=1 TO 8:FOR J=1 TO 8:A(I,J)=0:NEXT J,I

170 BOX (0,0)-(319,199)

180 BEEP:FOR I=0 TO 100:NEXT:BEEP
```

```
200 INPUTPEN X,Y

210 FOR J=1 TO 10:IF PTRIG=-1 GOTO 210
    ELSE NEXT J

220 IF X(0 OR Y(0 GOTO 200

230 IF X(33 GOTO 1000

240 C=INT((X-56)/24)+1:L=INT(Y/24)+1:
    CC=3*C+5:LL=3*L-2

250 IF C(1 OR L(1 OR L)8 GOTO 200

260 LOCATE CC,LL:PRINT GR$(A(C,L))

270 A(C,L)=-(A(C,L)=0)

280 GOTO 200


500 COLOR 1:FOR L=1 TO 8

510 Z=0

520 FOR C=1 TO 8

530 Z=2*Z+A(C,L)

540 NEXT C :Z(L,NG)=Z

550 LOCATE 36,3*L-2:PRINT USING "###";Z:BEEP

560 NEXT L

570 DEFGR$(2)= Z(1,NG),Z(2,NG),Z(3,NG),Z(4,NG),
               Z(5,NG),Z(6,NG),Z(7,NG),Z(8,NG)

580 BOX (0,160)-(32,192)

590 ATTRB 1,1:LOCATE 1,22:PRINT GR$(2):
    ATTRIB 0,0:LOCATE 5,22:PRINT GR$(2)

600 COLOR 4

610 GOTO 200
```

```
1000 V=1-(Y(33)-2*(Y )39 AND Y(73)
       -3*(Y)79 AND Y(113)

1010 ON V GOTO 200,500,30,2000


2000 CLS:LOCATE 13,0:PRINT "RECAPITULATION":
     LOCATE 20,24:PRINT "TAPEZ ";:COLOR 7, 1:
     PRINT "ENTREE";:COLOR 4,6:CONSOLE 2,22

2010 N1=-8:S=0

2020 LOCATE 0,3:N1=N1+8:N2=N1+7:
     IF N2)=NG THEN N2=NG:S=1

2030 FOR I=N1 TO N2

2040 DEFGR$(2)=Z(1,I),Z(2,I),Z(3,I),Z(4,I),
               Z(5,I),Z(6,I),Z(7,I),Z(8,I)

2050 PRINT USING "GR$(###)=";I;:
     FOR J=1 TO 8:PRINT Z(J,I);",";:NEXT J

2060 PRINT "    ";:COLOR 1:PRINT GR$(2):COLOR 4

2070 NEXT I

2080 R$=INPUT$(1):R=ASC(R$):IF R()13 GO TO 2080

2090 CLS:IF S=0 GO TO 2020

2100 CONSOLE 0,24:CLS
```

La première constatation après exécution du programme, est que la présentation des résultats laisse quelque peu à désirer sur le plan de l'esthétique.

Les résultats s'affichent correctement mais l'impression des Z(L,I) entre les virgules comportent trop d'espaces. Il y a un premier espace pour la place du signe (or nous savons que les

Z(L,I) sont toujours positifs. En outre, il y a un espace derrière le nombre affiché (c'est une propriété de l'instruction PRINT qui laisse toujours un espace derrière une valeur numérique -(M.Ref 87)-.

Par exemple le caractère "CROIX DE SAINT-ANDRE" qui a la forme d'un X donne la présentation suivante:

```
GR$(0)= 129 , 66 , 36 , 24 , 24 , 36 ,
 66 , 129 ,   X                          (avec le X Rouge)
```

Nous allons essayer d'améliorer la présentation en particulier en supprimant les espaces indésirables.

Plutôt que d'imprimer successivement les huit valeurs numériques caractérisant le caractère numéro I, nous allons bâtir une chaine de caractères composée des huit valeurs Z(L,I) (L variant de 1 à 8) transformées en chaines concaténées avec des caractères "virgule" les séparant.

Rappelons que l'opération de concaténation est la simple mise bout à bout de chaines (cette opération agissant sur les chaines de caractères est obtenue par l'opérateur d'addition (+) -(M.Ref 22)-.

Ainsi par exemple le programme:

```
10 A$ = "ABCD": B$= "EFGH"

20 C$= A$+ B$

30 PRINT C$
```

a pour effet l'affichage du résultat C$ de la concaténation de A$ et de B$, soit la chaine "ABCDEFGH".

Il nous faut d'abord transformer la valeur numérique Z(L,I) en une chaine de caractères par la fonction STR$ -(M.Ref 147)-, puis supprimer le premier caractère de cette chaine (réservé au signe qui, nous le savons, ne peut être qu'un + représenté par un espace, dans notre cas, inutile).

La fonction MID$ -(M.Ref 130)- permet d'extraire les caractères de la chaine à partir du second caractère et en conséquence supprimer le premier.

On prendra garde à ne pas confondre la fonction:

X$=MID$(Z$,P,Q)

qui permet d'obtenir en X$ les Q caractères de Z$ en commençant au caractère de rang P (si Q n'est pas précisé, MID$ (Z$,P) extrait tous les caractères de Z$ à partir du caractère de rang P) et la fonction: -(M.Ref 73)-

MID$(A$,P,Q)=Z$

qui dans la chaine A$ remplace les caractères à partir du caractère de rang P par les Q premiers caractères de Z$.

En conséquence nous transformons la valeur numérique Z(L,I) en une chaine ne comportant pas d'espace en première place par :

MID$(STR$(Z(L,I),2))

et la concaténons avec le caractère "," par

MID$(STR$(Z(L,I),2))+","

pour obtenir ce que nous appellerons une chaine élémentaire.

Il ne nous reste plus qu'à concaténer les différentes chaines élémentaires correspondant à L variant de 1 à 8.

On parvient ainsi à un programme de la forme

```
1510 FOR I=0 TO NG
1520 Z$=""
1530 FOR J=1 TO 8
1540 Z$=Z$+MID$(STR$(Z(J,I)),2)+","
1550 NEXT J
1560 Z$(I)=LEFT$(Z$,LEN(Z$)-1)
1570 NEXT I
```

La boucle 1510-1570 permet de bâtir la chaine Z$(I) pour toutes les valeurs de I variant de 0 à NG.

L'instruction 1520 met à "zéro" la chaine Z$ (Z$ est une chaine de longueur nulle ce qui est l'équivalent de zéro pour une valeur numérique).

La boucle 1530-1550 calcule dans Z$ le résultat de la concaténation des chaines élémentaires.

Enfin l'instruction 1560 place, dans Z$(I), la chaine Z$ dont on a supprimé le dernier caractère par l'opération LEFT$ -(M.Ref 127)-, puique l'on ne conserve que la partie gauche de Z$ sur une longueur égale à la longueur de Z$ (LEN(Z$)) -(M.Ref 128)- diminuée de 1. Cette opération permet de supprimer la dernière virgule provenant de la chaine élémentaire bâtie pour J = 8.

Le calcul de toutes les chaines Z$(I) demande un certain temps. Aussi est-il nécessaire de prévenir l'opérateur d'avoir à patienter. C'est l'objet de l'instruction 1500 que nous rajouterons en début de bloc :

```
1500 CLS:LOCATE 14,12:PRINT "UN INSTANT!"
```

qui affiche le message UN INSTANT! au milieu de l'écran.

* * * * * *

Il est toutefois nécessaire d'apporter quelques modifications supplémentaires au programme.

Il faut réserver des places mémoires pour le tableau Z$(I) soit 127 places, par l'instruction DIM. Nous complèterons donc la deuxième instruction de l'étiquette 20 :

```
20 DEFINT A-Z:DIM A(8,8),Z(8,127),Z$(127):NG=-1
```

Par ailleurs il nous faut réserver de la place mémoire pour le traitement des chaines de caractères imposé par les instructions 1510 à 1570.

Cette réservation se fait par l'instruction CLEAR -(M.Ref 32)-. C'est le premier paramètre de cette instruction qui définit le nombre d'octets réservés au calcul des chaines de caractères.

Nous avions utilisé l'instruction CLEAR à l'étiquette 10 sous la forme :

```
10 CLEAR ,,3
```

Le premier paramètre n'ayant pas été précisé, c'est la valeur 300 qui avait été retenue par défaut.

Il n'est pas toujours facile d'estimer quelle place mémoire peut être nécessaire. Aussi proposons-nous la méthode "expérimentale" suivante.

Nous pourrions essayer le programme en mettant a priori une valeur arbitraire comme premier paramètre pour CLEAR, par exemple 5000, puis bâtir 128 caractères et vérifier que cela "passe"

Il est plus commode de "court-circuiter" la partie du programme permettant de bâtir les caractères (de 30 à 1010) et de bâtir un tableau de 8 x 128 valeurs des Z(J,I) avec des valeurs aléatoires (comprises entre 0 et 255) afin de tester les instructions 1500 à 1570.

A cet effet nous allons rajouter les instructions provisoires suivantes:

```
24 NG=127:DS=256

25 GOTO 1200

1200 FOR I=0 TO 127:COLOR 1:PRINT I;" ";:COLOR 4

1210 FOR J=1 TO 8

1220 Z(J,I)=INT(DS*RND)

1230 PRINT Z(J,I);",";

1240 NEXT J

1250 PRINT:NEXT I
```

Par ailleurs nous remplaçons l'instruction 10 par :

```
10 CLEAR 5000,,3
```

Enfin nous rajoutons les instructions suivantes dans le bloc 1500-1570 :

```
1525 COLOR 1:PRINT I;"   ";:COLOR 4

1555 PRINT FRE(X$)
```

L'instruction 24 fixe à 127 la valeur de NG (cela nous sera utile lors de l'affichage des résultats (bloc 2000-2090).

En outre on pose DS = 256, constante utilisée à l'étiquette 1220 (il est préférable d'utiliser une constante sous forme d'une variable plutôt que sous forme d'une valeur numérique dans une formule. Cela gagne du temps au moment de l'éxécution car l'interpréteur BASIC n'a pas à traduire cette valeur numérique. Cela est encore plus important lorsque la formule figure à l'intérieur d'une boucle).

L'instruction 25 branche sur 1200 en court-circuitant tout le programme de création de caractères.

Les instructions 1200 à 1250 permettent de créer le tableau des Z(J,I).

*** 1200
a/ Ouvre une boucle en I.

b/ et c/ Affiche I en Rouge (1).

Cela permettra de suivre pas à pas la création du tableau.

d/ Remet la couleur des caractères en Bleu (4).

Ces petits détails de changements de couleurs peuvent paraitre superflus. Ils sont en réalité très commodes pour pouvoir suivre aisément le déroulement du programme.

*** 1210
Ouvre une boucle en J.

*** 1220
La fonction RND -(M.Ref 137)- calcule à chaque appel un nombre pseudo-aléatoire compris entre 0 et 1 (0 inclus et 1 exclus).

Ainsi INT(DS*RND) calcule à chaque appel un nombre entier pseudo-aléatoire compris entre 0 et 255 (DS = 256).
C'est cette valeur toujours différente d'un pas de calcul à l'autre que nous chargeons dans Z(J,I).

*** 1230
Affiche la valeur de Z(J,I) suivie d'une virgule et sans saut à la ligne (point-virgule en fin d'instruction).

*** 1240
Ferme la boucle en J.

*** 1250
a/ Provoque un saut à la ligne (fin des huit valeurs de Z(J,I) pour J de 1 à 8).

b/Ferme la boucle en I.

Ainsi ce bloc d'instructions de 1200 à 1250 fabrique un tableau des Z(J,I), rempli de valeurs aléatoires comprises entre 0 et 255, ce qui simule un tableau réél mais beaucoup plus rapide à constituer.

*** 1525
Affiche au cours de la constitution des variables Z$(I) la valeur de I en Rouge, puis ramène la couleur des caractères au Bleu. Cette instruction permettra de suivre pas à pas l'évolution du calcul de la variable Z$(I).

*** 1555
La fonction FRE(X$) -(M.Ref 121)- donne la place mémoire restant disponible pour le traitement des chaines de caractères.

L'instruction 1255 nous permettra de suivre, en fonction de l'indice I, l'évolution de la place disponible et donc d'évaluer la place restante en fin d'éxécution du bloc 1500-1570.

Il nous sera alors possible d'ajuster la valeur du premier paramètre R de 10 CLEAR R,,3.

L'expérience (qu'il est interessant d'exécuter réellement) montre qu'une valeur de l'ordre de 4000 est suffisante. C'est elle que nous retiendrons tout en notant que ce programme n'étant pas très gourmand en mémoire, nous pouvons nous permettre de ne pas être trop avares en place mémoire. Mais il est bon de retenir la méthode employée pour pouvoir serrer au plus près les valeurs nécessaires dans des cas plus critiques.

* * * * * *

Il est enfin nécessaire de modifier quelque peu le bloc 2000-2090 d'affichage des résultats.

Les instructions ajoutées (ou modifiées) au programme donné ci-dessus figurent précédées des symboles +++ et les instructions provisoires précédées des symboles Prv.

```
+++   10 CLEAR 4000,,3

+++   20 DEFINT A-Z:DIM A(8,8),Z(8,127),
         Z$(127):NG= -1

Prv   24 NG=127:DS=255

Prv   25 GOTO 1200
```

Les instructions de 30 à 1010 sont inchangées et sautées. Nous ne les reproduisons pas ici.

```
Prv 1200 FOR I=0 TO 127:COLOR 1:PRINT I;" ";:
         COLOR 4

Prv 1210 FOR J=1 TO 8

Prv 1220 Z(I,J)=INT(DS*RND)

Prv 1230 PRINT Z(I,J);",";

Prv 1240 NEXT J

Prv 1250 PRINT:NEXT I
```

```
+++ 1500 CLS:LOCATE 14,12:PRINT "UN INSTANT !"

+++ 1510 FOR I=0 TO NG

+++ 1520 Z$=""

Prv 1525 COLOR 1:PRINT I;"  ";:COLOR 4

+++ 1530 FOR J=1 TO 8

+++ 1540 Z$=Z$+MID$(STR$(Z(J,I)),2)+","

+++ 1550 NEXT J

Prv 1555 PRINT FRE(X$)

+++ 1560 Z$(I)=LEFT$(Z$,LEN(Z$)-1)

+++ 1570 NEXT I

+++ 2000 CLS:LOCATE 13,0:PRINT "RECAPITULATION":
         LOCATE 20,24:PRINT "TAPEZ ";:COLOR 7, 1:
         PRINT "ENTREE";:COLOR 4,6:CONSOLE 3,22

    2010 N1=-8:S=0

+++ 2020 LOCATE 0,5:N1=N1+8:N2=N1+7:
         IF N2)=NG THEN N2=NG:S=1

    2030 FOR I=N1 TO N2

    2040 DEFGR$(2)=Z(1,I),Z(2,I),Z(3,I),Z(4,I),
                   Z(5,I),Z(6,I),Z(7,I),Z(8,I)

+++ 2050 COLOR 0:PRINT USING "GR$(###)=";I;:
         COLOR 1:PRINT GR$(2);"  ";:
         COLOR 4:PRINT Z$(I):PRINT

+++ 2060 NEXT I

+++ 2070 R$=INPUT$(1):R=ASC(R$):
         IF R()13 GO TO 2070

+++ 2080 CLS:IF S=0 GO TO 2020

+++ 2090 CONSOLE 0,24:CLS
```

Les modifications du bloc 2000-2090 sont évidentes.

L'instruction 2050 permet d'afficher la valeur du compteur I de caractères en Noir, le caractère en Rouge et enfin la liste des huit valeurs caractéristiques séparées par des virgules en Bleu par l'affichage de la variable Z$(I) precédemment calculée.

Il est intéressant de faire tourner ce programme jusqu'au bout, ne serait-ce que pour admirer lors de la récapitulation les superbes caractères aléatoires fabriqués (ils ont un petit aspect rappelant le chinois !).

* * * * * *

Il nous reste maintenant à ajouter le bloc d'instructions permettant d'obtenir la récapitulation des résultats sur l'imprimante.

Bien entendu, il faut laisser à l'utilisateur la possibilité de décider s'il désire ou non cette impression.

A cet effet, nous plaçons un message à l'écran après effacement des derniers résultats récapitulés proposant l'impression ou la clôture du programme.

Les instructions du bloc d'impression sont numérotées de 3000 à 5000 et placées après le bloc des instructions de récapitulation.

```
3000 LOCATE 0,12:
     PRINT "Pour imprimer les resultats tapez ";:
     COLOR 0,1:PRINT "ENTREE":
     COLOR 4,6:LOCATE 0,14:
     PRINT "Pour cloturer                tapez ";:
     COLOR 0,1:PRINT "F":COLOR 4,6

3010 LOCATE 20,16:R$=INPUT$(1)

3020 ON 1-(ASC(R$)=13)-2*(R$="F")
     GOTO 3010,3500,4000

3500 OPEN "O",1,"LPRT:(80)"
```

```
3510 GOSUB 5000

3520 FOR I=0 TO NG

3530 I$=STR$(I):I$=RIGHT$(I$,LEN(I$)-1)

3540 PRINT #1,"DEFGR$(";I$;") = ";Z$(I)

3550 NEXT I

3560 GOSUB 5000

4000 CLS:END

5000 PRINT #1:PRINT #1:FOR I=1 TO 80:
     PRINT #1;"-";:NEXT:PRINT #1:PRINT #1:RETURN
```

*** 3000
Place en lignes 12 et 14 le message:

Pour imprimer les résultats	tapez	ENTREE
Pour clôturer	tapez	F

avec les deux mots ENTREE et F en Noir sur fond Rouge pour bien faire comprendre qu'il s'agit des touches ENTREE et F.

*** 3010
Le curseur (rendu apparent par la valeur 1 du troisième paramètre de LOCATE se place au centre de le ligne 16 et le programme s'arrête en attendant la frappe d'une touche (instruction INPUT$(n) -(M.Ref 174)- qui place dans la variable R$ le caractère correspondant à la touche frappée).

On notera qu'un seul caractère est attendu, le paramètre n figurant dans la parenthèse de INPUT$ étant égal à 1.

*** 3020
L'expression

```
1 - (ASC(R$)=13) - 2*(R$="F")
```

prend les valeurs:

1 si la touche frappée n'est ni la touche ENTREE ni la touche F

2 si la touche frappée est la touche ENTREE dont le numéro de code ASCII est 13, (ASC(R$)= 13) étant alors égal à -1.

3 si la touche frappée est la touche F, (R$= "F") étant égal à -1 (voir l'utilisation des variables logiques dans ce même programme aux instructions 1000 et 1010).

Ainsi le programme est branché sur l'étiquette 3500 si la touche ENTREE a été frappée, sur l'étiquette 4000 si la touche F a été frappée.

Nous placerons donc le bloc d'instructions d'impression à partir de 3500 et l'instruction de clôture en 4000.

Il est recommandé de lire l'Annexe II consacrée à l'utilisation des imprimantes avant d'aborder la suite de ce chapitre.

*** 3500
L'instruction OPEN -(M.Ref 177)- affecte le canel numéroté 1, à la sortie sur imprimante.

Le numéro de canal est arbitraire (il doit être compris entre 1 et 16 et est leissé au choix du programmeur).

La lettre O entre guillemets (3) précise que ce canal sera utilisé en sortie (O pour OUTPUT), c'est-à-dire dans le sens ordinateur vers périphérique, ici ordinateur vers imprimante.

L'affectation de ce canal à l'imprimante est précisée par le mot clé LPRT:.

Dans notre cas nous avons choisi d'imprimer des lignes contenant au plus 80 caractères d'où la valeur 80 entre parenthèses (ceci suppose que nous utilisons l'imprimante à aiguilles PR 90-080).

On notera soigneusement la syntexe de l'instruction. Le 0 doit être placé entre guillemets, le numéro du canal entre deux virgules. Ne pas oublier les deux-points (:) après LPRT ni les guillemets.

Le numéro de canal de sortie des données sur imprimante étant ainsi fixé, tous les ordres d'impression seront de la forme:

PRINT #1 (chaines, variables, etc..)

(1 étant le numéro de canal). Il est possible d'entremêler des instructions d'affichage à l'écren avec des instructions d'impression sur l'imprimante. Ces ordres d'affichage à l'écran sont alors des ordres PRINT simples, sans numéro de canal.

*** 3510
Branche sur le sous-programme 5000 qui permet

a/ et b/ O'effectuer deux sauts à la ligne sans impression.

c/ d/ et e/ Par une boucle de 80 pas, d'imprimer une ligne complète de 80 tirets (-).

f/ et g/ O'effectuer encore deux sauts à le ligne.

*** 3520
Ouvre une boucle en I de 0 à NG pour l'impression des NG +1 résultats.

*** 3530
Calcule une chaine de ceractères I$ représentant I, en supprimant l'espace remplaçant le signe + et contenant le nombre exact de digits de I.

*** 3540
Imprime la chaine "DEFGR$ (", suivie la chaine I$, de la chaine ") = " et enfin de la chaine Z$ (I) calculée en 1500-1570 et déjà utilisée en 2050.

Ainsi le caractère CROIX DE SAINT-ANDRE déjà cité, placé en numéro 21 apparait sous la forme:

DEFGR$ (21) = 129, 66, 36, 24, 24, 36, 66,129

*** 3550
Ferme la boucle en I.

*** 3560
Branche sur le sous-programme 5000 (voir 3510)

*** 4000
a/ Efface l'écran.

b/ L'instruction END est indispensable pour éviter que le sous-programme en 5000 ne soit éxécuté directement après l'instruction 4000, ce qui provoquerait l'erreur 3 (RG, c'est-à-dire RETURN WITHOUT GOSUB) (l'instruction a été rencontrée sans avoir été précédée d'un GOSUB).

Rappelons qu'une instruction END est toujours nécessaire si le programme doit s'arrêter à une instruction qui n'est pas plecée à la dernière étiquette.

Enfin l'instruction END ferme le cenal 1 ouvert en 3500. Cela évite l'instruction CLOSE 1 -(M.Ref 170)- de fermeture du canal.

6

Clavier

Nous nous proposons d'éteblir un programme permettant de tester les différentes touches du clavier. Cela nous conduira, en particulier, à voir comment identifier la dernière touche frappée.

Nous dessinerons donc sur l'écran un ensemble de cases rectangulaires reproduisent du mieux possible la forme et la position des différentes touches disponibles sur le clevier. La frappe d'une touche particulière se traduira par l'inscription, sur la "touche écran", du caractère ou du symbole correspondant.

Les deux touches CNT et MAJUSCULE (cette dernière est marquée d'un point jaune), n'eyant pas de signification en elles-mêmes, puisqu'elles ne font que modifier la "veleur" de la touche frappée en même temps, ne seront pas représentées.

Nous décidons, a priori, de trecer le symbole ou le caractère qui vient d'être frappé au clavier, en couleur Rouge, de façon à bien le distiguer des caractères déjà frappés qui seront maintenus sur l'écren, en Bleu, si ce sont des caractères alphanumériques, en Noir si ce sont les touches de fonction (STOP, ACC, Flèches, etc) et enfin en Blanc sur fond Magenta si ce sont des caractères "controle" (obtenus en frappent une touche alphabétique tout en eppuyant sur la touche CNT)

Nous aurons toutefois à prendre garde à ce que les différentes touches de fonction ne sont en réalité que des touches "controle" et qu'il nous faudra non seulement afficher à l'écran la touche fonction mais également la touche "controle" correspondante.

Commençons par le dessin des touches.

```
10 CLS:SCREEN 0,6,6:CLEAR ,,5:LOCATE 0,0,0

20 Y0=14:DY=26:BOX (4,Y0)-(33,Y0+DY):
   BOX (36,Y0)-(260,Y0+DY):
   BOX (268,Y0)-(284,Y0+DY):
   BOX (290,Y0)-(307,Y0+DY)

30 S=12:FOR I=4 TO 15:GOSUB 2000:NEXT

40 Y0=54:BOX (44,Y0)-(236,Y0+DY):
   BOX (268,Y0)-(284,Y0+DY):
   BOX (290,Y0)-(307,Y0+DY)

50 S=4:FOR I=4 TO 14:GOSUB 2000:NEXT

60 Y0=94:BOX (4,Y0)-(33,Y0+DY):
   BOX (52,Y0)-(212,Y0+DY):
   BOX (215,Y0)-(264,Y0+DY):
   BOX (268,Y0)-(307,Y0+DY)

70 S=12:FOR I=5 TO 13:GOSUB 2000:NEXT

80 Y0=134:BOX (60,Y0)-(204,Y0+DY):
   BOX (268,Y0)-(286,Y0+DY):
   BOX (288,Y0)-(307,Y0+DY)

90 S=4:FOR I=5 TO 13:GOSUB 2000:NEXT

100 BOX (71,174)-(200,199)

2000 X=16*I-S:LINE (X,Y0)-(X,Y0+DY):RETURN
```

*** 10
a/ Efface l'écran

b/ Fixe au Noir (0) la couleur des cerectères, au Cyan (6) les couleurs de fond et d'entourage écran.

c/ Réserve 5 places mémoires pour les caractères graphiques (nous en aurons besoin pour représenter les touches INS, EFF et la touche "Flèche Retour").

On notera que l'instruction CLEAR -(M.Ref 32)- est placée en tout début de programme. Elle provoque en effet, entre autres, le mise à zéro de toutes les mémoires. On notera également les deux virgules précédant le chiffre 5. Les deux valeurs numériques, non précisées, avant la première virgule et entre les deux virgules sont destinées à la réservation des places mémoires pour le traitement des chaines de caractères (300 places retenues par défaut) et pour le logement de progremmes ou de données en binaire (voir le programme MINICARACTERES pour cette dernière réservation d'espace).

d/ Place le curseur dans le coin haut-geuche de l'écran (0,0)et l'efface (effet du troisième zéro).

*** 20

a/ Fixe la valeur de l'ordonnée YO de la ligne supérieure de la première rangée de "touches"

b/ Fixe le valeur de la heuteur DY des touches.

Les valeurs de DY et de YO ont été choisies "expérimentalement", de façon à obtenir une présentation harmonieuse.

Les lignes supérieure et inférieure des "touches" sont alors placées de telle sorte que l'on peut écrire deux caractères l'un au dessus de l'autre, séparés par un blanc sans déborder sur le cadre (DY = 3 x 8 + 2).

c/ La première touche tracée par l'instruction BOX -(M.Ref 187)- est destinée à la touche de fonction STOP.

On notere que la couleur n'a pas été précisée en fin d'instruction. C'est donc la couleur caractère Noir (0), fixée en 10 par l'instruction SCREEN qui est retenue par défaut.

d/ L'instruction BOX suivante trace un cadre englobant l'ensemble de la première rangée depuis la touche 1 jusqu'à la touche de fonction ACC.

Il sera nécessaire de tracer ensuite les séparations entre touches (voir étiquette 30).

e/ et f/ Tracé des deux touches Flèche Haut et Flèche Bas à droite de la première rangée.

*** 30
Les instructions regroupées sous cette étiquette trecent des traits verticaux de séparation des touches d'une même rangée.

a/ Le paramètre S permet de modifier l'abscisse des traits verticaux d'une rangée de touches à l'autre.

Ces abscisses ont été choisies de telle sorte que l'on puisse inscrire un caractère centré entre deux traits (l'abscisse du "point milieu" d'un caractère est un multiple de huit plus 3.5).

b/ Ouverture d'une boucle en I permettant de tracer les 12 traits verticaux de la première rangée.

c/ Appel au sous-programme 2000 de tracé d'un trait vertical.

d/ Fermeture de la boucle.

*** 40-50, 60-70, 80-90
Ces trois groupes de deux étiquettes sont consacrées au tracé des trois autres rangées de touches.

*** 100
Tracé du cadre rectangulaire représentent la barre d'espece.

L'ebscisse des points de gauche est un multiple de 8 diminué de 1 et celle des points de droite un multiple de 8.

Les points du cadre n'appartiennent donc pas aux blocs de huit points, liés per la couleur, de l'abscisse 72 à l'abscisse 199. On pourra donc modifier le couleur à l'intérieur du cadre sans changer celle du cadre lui-même.

*** 2000
a/ Calcul de l'abscisse X du trait vertical de rang I.

Les treits verticaux sont donc espacés de 16 points écran. Le paramètre S permet de décaler les diverses rangées de touches suivant leur position en hauteur sur l'écran (on remarquera que le clavier du TO7 a ses touches disposées en quinconce).

b/ Trace une ligne verticele d'abscisse X entre les points d'ordonnées YO et YO + OY.

c/ Instruction de fin de sous-progremme -(M.Ref 57)-

On peut se demender pourquoi les instructions 20, 40, 60 et 80 commencent par fixer la valeur de YO, utilisée uniquement dans l'étiquette elle-même. Il aurait été possible en effet de remplacer YO par sa valeur dans les trois ou quatre instructions BOX qui y figurent.

L'intérêt d'utiliser YO, au lieu de la valeur numérique, n'est, dans ce cas, ni un gain dans le temps d'exécution, ni un gein de plece mémoire. C'est par contre une commodité lors de la mise au point du "dessin" du clavier à l'écran. Il est plus commode de chenger, en une seule fois, le valeur d'un paramètre en modifiant la valeur de YO, plutôt que de changer plusieurs valeurs numériques à l'intérieur des diverses instructions BOX, avec le risque d'en oublier une. Or il faut savoir que la conception du "dessin" ne se fait pas ex abrupto. Un certain nombre d'essais est nécesseire avant d'obtenir un résultat satisfaisant evec, entre autres, quelques changements sur la position des rangées donc sur les valeurs des YO.

Une fois tapées les instructions 10 à 100, il est recommandé d'essayer le programme pour relever les erreurs de syntaxe et juger l'effet produit par le grephisme obtenu.

Ce peut être l'occasion de modifier la disposition de quelques "touches" si nous constatons que le dessin ne se présente pas aussi bien que nous l'avions imaginé sur le papier. Il est en effet préférable de partir d'un dessin quasi définitif avant de rentrer les données colonne-ligne correspondant à chaque caractère. Cela évitera des modifications ultérieures de données, toujours fastidieuses.

On constatera que les touches de fonction INS et EFF bâties par l'instruction BO sont trop étroites pour contenir trois caractères. Leurs formes et leurs positions ont été imposées par la forme et la position des touches "Flèches" des rangées supérieures.

Il est donc nécessaire de bâtir des caractères spéciaux dits "graphismes utilisateur" -(M.Ref 190)- pour y inscrire les mentions INS et EFF sur appel de ces touches.

Les caractères de INS seront représentés par un graphisme reproduisant les deux lettres I et N dans un même cerré de 8 x 8 points et un graphisme représentant un S seul. Ainsi INS n'occupera que deux cases caractère.

On sait qu'un graphisme utilisateur occupe, comme un caractère typographique une case 8 x 8 points écran, l'abscisse de la rangée verticale des huit points de gauche étant un multiple de huit et celle de la rangée verticale des huit points de droite étant le même multiple de huit eugmenté de sept.

L'abscisse des points de droite du cadre représentant la touche INS est 286. Ces points se trouvent donc à l'intérieur de la "case" caractère dans la colonne 35 (286 / 8 = 35.75). Si l'on veut écrire un caractère, en l'occurrence le S de INS, en colonne 35 (ligne 18), il est donc nécessaire de recréer la ligne verticale des huit points de droite dans le caractère lui-même.
Le S du second caractère de INS sera donc complèté par une barre verticale de huit points destinés à rétablir la continuité du cadre à droite après affichege du caractère. Ces huit points seront placés dans la septième colonne du caractère (l'abscisse du coin geuche d'un caractère plecé en colonne 35 par un LOCATE 35,Y est 35 x 8 = 280; l'ebscisse du septième point du caractère est donc 280 + (7-1) = 286).

Nous bâtirons donc pour INS les deux ceractères ayant les formes suivantes :

```
000.0..0        ..000.0.
.0..0..0        .0....0.
.0..0..0        .0....0.
.0..00.0        ..0...0.
.0..0.00        ...0..0.
.0..0..0        ....0.0.
.0..0..0        ....0.0.
000.0..0        .000..0.
```

Un raisonnement enelogue nous conduit à bâtir les deux caractères suivants pour la touche EFF (dans ce ces c'est le premier caractère E qui doit être complèté à gauche par une barre verticale rétablissant le cadre après affichage du caractère).

```
0..000..        000..000
0..0....        0....0..
0..0....        0....0..
0..00...        00...00.
0..0....        0....0..
0..0....        0....0..
0..0....        0....0..
0..000..        0....0..
```

Il nous reste enfin à bâtir le graphisme reproduisant le forme du symbole de la "Flèche Retour" et les graphismes représentant les quetre Flèches Heut, Bas, Geuche et Droite.

Pour ces dernières nous avons vu dans le premier programme que nous pouvons les obtenir en concaténant le ceractère CHR$(22) (touche ACC) evec les caractères "-" (Haut), "/" (Bas), "," (Gauche) et "." (Droit).

Pour la Flèche Retour nous bâtirons le ceractère suivant :

```
..0.....
.0......
00000000
.0.....0
..0....0
.......0
.......0
00000000
```

Bien entendu nous utiliserons le programme du chapitre précédent -(M.Bas 63)- pour bâtir ces cinq graphismes.

Les différentes touches seront identifiées par leur numéro de code ASCII. A chaque numéro de code il nous faut faire correspondre les numéros de ligne et de colonne écran où nous afficherons le symbole correspondant.

Nous utiliserons pour ce faire deux variables C(N) et L(N), représentant la ligne et la colonne d'affichage du symbole de numéro de code N. Les caractères "controle" seront placés dans la même case que le caractère alphabétique correspondant, mais deux lignes au dessus.

Nous noterons enfin que les touches de fonction ne sont que des caractères "controle".

La frappe d'une touche de fonction ne peut ainsi se différencier de la frappe du caractère "controle" correspondant. Nous aurons donc deux affichages à faire dans ce cas, l'affichage de la touche de fonction et l'affichage du caractère alphabétique correspondant au carectère "controle" avec le convention de couleur retenue pour ce type de caractère.

Rappelons que les caractères "controle" correspondant aux différentes touches de fonction sont les suivantes :

```
STOP              02      CNT B
FLECHE GAUCHE     08      CNT H
FLECHE DROITE     09      CNT I
FLECHE BAS        10      CNT J
FLECHE HAUT       11      CNT K
RAZ               12      CNT L
ENTREE            13      CNT M
ACC               22      CNT V
INS               28      )   il n'y a pas de
DEL               29      ) caractère "controle"
FLECHE RETOUR     30      )    correspondant
```

Toutes ces données, indispensables pour la suite du programme, sont stockées par les instructions suivantes :

```
150 DEFGR$(0) = 32,64,255,65,33,1,1,255:
    DEFGR$(1) = 156,144,144,152,144,144,144,156:
    DEFGR$(2) = 231,132,132,198,132,132,132,132:
    DEFGR$(3) = 233,73,73,77,75,73,73,233:
    DEFGR$(4) = 58,66,66,34,18,10,10,114

200 DATA 5,2,7,2,9,2,11,2,13,2,15,2,17,2,19,2,
         21,2,28,9,27,4,20,19,25,4,22,19,26,9

210 DATA 23,4,5,4,7,4,9,4,11,4,13,4,15,4,17,4,
         19,4,21,4

220 DATA 28,7,27,2,20,17,25,2,22,17,26,7,24,19

230 DATA 6,9,16,19,12,19,11,14,10,9,13,14,15,
         14,17,14,20,9,19,14,21,14,23,14

240 DATA 25,14,18,19,22,9,24,9,7,14,12,9,9,14,
         14,9,18,9,14,19,8,19,10,19,16,9,8,9,
         0,0,0,0,0,0,24,17

250 DIM C(94),L(94)

260 FOR I=33 TO 94:READ C(I),L(I):NEXT

270 RESTORE 230:FOR I=1 TO 26:
    READ C(I),L:L(I)=L-2:NEXT

280 A$=CHR$(22):FG$=A$+",":FD$=A$+".":
    FH$=A$+"-":FB$=A$+"/"
```

*** 150
Définition des cinq graphismes utilisateur.

GR$(0) est la Flèche Retour. GR$(1) et GR$(2) sont destinés à représenter la touche EFF, GR$(3) et GR$(4) la touche INS.

*** 200
Données colonne, ligne pour les caractères de numéros de code ASCII compris entre 33 et 47 (ponctuation).

*** 210
Données colonne, ligne pour les caractères de numéros de code ASCII compris entre 48 et 57 (chiffres).

*** 220
Données colonne, ligne pour les caractères de numéros de code ASCII compris entre 58 et 64 (ponctuation).

*** 230
Données colonne, ligne pour les caractères de numéros de code ASCII compris entre 65 et 76 (lettres majuscules de A à L).

*** 240
Données colonne, ligne pour les caractères de numéros de code ASCII compris entre 77 et 94 (lettres majuscules de M à Z (90) et symbole V renversé (94); les codes 91 à 93 ne correspondent pas à une touche du clavier, ce qui explique les six zéros en fin de DATA).

*** 250
Réserve 95 places mémoire (0 à 94) pour les deux variables C(I) et L(I) -(M.Ref 46)-

*** 260
Boucle de lecture des données et de chargement dans les veriables C(I) et L(I), pour les caractères de numéros de code ASCII compris entre 33 et 94, c'est à dire pour l'ensemble des touches exceptées les touches de fonction et les touches appelées avec la touche CNT enfoncée (caractères "controle").

*** 270
Un caractère "controle" est présenté à l'écran dans la même colonne que le ceractère alphebétique correspondant, mais deux lignes au dessus. La varieble C(I) est donc la même et la verieble L(I) est diminuée de 2. Il suffit donc de relire les données correspondant aux ceractères elphebétiques et de réduire de 2 les données correspondant à la variable L(I).
Le relecture des données pour les carectères elphabétiques impose de remettre le pointeur de DATA au début de ces données. C'est le rôle de l'instruction RESTORE 230 -(M.Ref 100)-

*** 280
Construction des quatre caractères Flèches par conceténetion du ceractère CHR$(22) (Touche ACC) avec les caractères "," (Flèche Gauche), "." (Flèche Droite), "-" (Flèche Haut) et "/" (Flèche Bas).

Il nous reste maintenant à bâtir le programme principal permettent d'efficher à l'écran le symbole correspondant à la touche frappée.

Rappelons que nous evons décidé d'afficher le symbole correspondant à le dernière touche frappée en Rouge (1) pour bien le distinguer des autres symboles. Oès qu'une autre touche a été frappée, la couleur du symbole doit passer du Rouge au Bleu (4), si c'est un ceractère alphabétique, du Rouge au Blanc (7) sur fond Magenta (5), si c'est un caractère "controle", du Rouge au Noir (0), si c'est une touche de fonction.

Enfin dans le cas des touches de fonction, le caractère "controle" correspondant doit s'afficher. Oe même sur frappe d'un caractère "controle" correspondant à une touche de fonction, la touche de fonction doit s'afficher.

```
300 COLOR 4:K=-1:LOCATE 5,0,0:
    PRINT "TESTEZ LES DIFFERENTES TOUCHES"

310 R$=INPUT$(1)

320 R=ASC(R$):IF R)96 THEN CLS:SCREEN 4,6,6:END
    ELSE IF R(27 THEN RN$=CHR$(R+64) ELSE RN$=R$

330 IF K=-1 GOTO 370

340 IF K(27 THEN C1=7:C2=5 ELSE C1=4:C2=6

350 IF K)26 AND K(33 GOTO 370

360 LOCATE C(K),L(K):COLOR C1,C2:PRINT RA$

370 K=R:IF K)26 AND K(33 GOTO 390

380 LOCATE C(K),L(K):COLOR 1,6:
    PRINT RN$:RA$=RN$

390 ON 1-(K=2)-2*(K=8)-3*(K=9)-4*(K=10)
    -5*(K=11)-6*(K=12)-7*(K=13)-8*(K=22)
    -9*(K=28)-10*(K=29)-11*(K=30)-12*(K=32)
    GOTO 310,400,500,600,700,800,900,
    1000,1100,1200,1300,1400,1500
```

```
400 LOCATE 1,3:COLOR 1,6:PRINT "STP"

410 R$=INPUT$(1):LOCATE 1,3:
    COLOR 0,6:PRINT "STP":GOTO 320

500 LOCATE 34,8:COLOR 1,6:PRINT FG$

510 R$=INPUT$(1):LOCATE 34,8:
    COLOR 0,6:PRINT FG$:GOTO 320

600 LOCATE 37,8:COLOR 1,6:PRINT FD$

610 R$=INPUT$(1):LOCATE 37,8:
    COLOR 0,6:PRINT FD$:GOTO 320

700 LOCATE 37,3:COLOR 1,6:PRINT FB$

710 R$=INPUT$(1):LOCATE 37,3:
    COLOR 0,6:PRINT FD$:GOTO 320

800 LOCATE 34,3:COLOR 1,6:PRINT FH$

810 R$=INPUT$(1):LOCATE 34,3:
    COLOR 0,6:PRINT FH$:GOTO 320

900 LOCATE 1,13:COLOR 1,6:PRINT "RAZ"

910 R$=INPUT$(1):LOCATE 1,13:
    COLOR 0,6:PRINT "RAZ":GOTO 320

1000 LOCATE 27,13:COLOR 1,6:
     PRINT "ENTREE"

1010 R$=INPUT$(1):LOCATE 27,13:
     COLOR 0,6:PRINT "ENTREE":GOTO 320

1100 LOCATE 29,3:COLOR 1,6:PRINT "ACC"

1110 R$=INPUT$(1):LOCATE 29,3:
     COLOR 0,6:PRINT "ACC":GOTO 320

1200 LOCATE 34,18:COLOR 1,6:
     PRINT GR$(3);GR$(4)

1210 R$=INPUT$(1):LOCATE 34,18:
     COLOR 0,6:PRINT GR$(3);GR$(4):GOTO 320
```

```
1300 LOCATE 36,18:COLOR 1,6:
     PRINT GR$(1);GR$(2)

1310 R$=INPUT$(1):LOCATE 36,18:
     COLOR 0,6:PRINT GR$(1);GR$(2):GOTO 320


1400 LOCATE 35,13:COLOR 1,6:
     ATTRB 1,0:PRINT GR$(0)

1410 R$=INPUT$(1):LOCATE 35,13:
     COLOR 0,6:PRINT GR$(0):ATTRB 0,0:
     GOTO 320


1500 BOXF(72,175)-(199,198),-2

1510 R$=INPUT$(1):
     BOXF(72,175)-(199,198),-6:GOTO 320
```

*** 300

a/ Fixe la couleur des caractères au Bleu (4).

b/ Fixe à -1 la valeur de la variable K.

K sera utilisé comme un indicateur pour signaler le début du programme. On sera amené en effet à changer la couleur du dernier symbole affiché. Or en début de programme il n'y a pas de symbole précédent.

Il est donc nécessaire, en début de programme, de sauter la partie de programme (340 - 360) consacré au changement de couleur (voir l'instruction sous l'étiquette 330).

c/ Place le curseur en ligne 0, colonne 5 et l'efface.

d/ L'affichage du message "TESTEZ LES DIFFERENTES TOUCHES", avertit l'utilisateur que le test est prêt. La lecture des données (150 - 280) prend en effet un certain temps pendant lequel la frappe des touches est sans effet.

*** 310

Le programme se met en attente de la frappe d'un caractère (et d'un seul puisque le paramètre N figurant dans le INPUT$(N) -(M.Ref 174)- est égal à 1) et le place dans la variable R$.

*** 320

a/ R est le numéro de code ASCII du caractère figurant dans R$ -(M.Ref 111)-

b/ Il est impossible de stopper le programme par intervention sur la touche STOP ou en tapant CNT C, puisque chacun de ces caractères est interprété comme l'un des caractères à afficher à sa place sur l'écran.

Par contre les caractères minuscules peuvent être utilisés comme signal de clôture de programme, puisqu'ils n'ont pas à être affichés sur l'écran.
Le numéro de code ASCII de "a" est 97. Si par suite c'est une minuscule qui est tapée, R est supérieur à 96, ce qui est utilisé pour brancher le programme sur les instructions CLS (effacement de l'écran), SCREEN 4,6,6 (remise des couleurs à leurs valeurs standard et END (fin de programme).

Si une touche quelconque, autre qu'un caractère minuscule est frappé ce sont les instructions situées après ELSE qui sont exécutées.

Ces instructions comportent elles aussi une condition.

Si le numéro de code ASCII R de R$ est inférieur à 27, c'est que le caractère est un caractère "controle". Or à un ceractère "controle" ne correspond aucun symbole affichable.

Il est donc nécessaire d'afficher le caractère alphabétique, tapé avec la touche CNT maintenue, en lieu et place du caractère "controle" lui-même.

Le numéro de code ASCII de CNT A est 1 et le numéro de code de A est 65. Par suite le caractère alphabétique, correspondant au caractère "controle" de numéro de code R, a R + 64 pour numéro de code propre.
En conséquence si R est inférieur à 27, on affichera le caractère RN$ = CHR$(R+64) sinon (ELSE) on affichera le caractère R$ lui-même (RN$ = R$).

*** 330

Si K est égal à -1, ce qui n'arrive qu'en début de programme, les instructions 340 à 360, consacrées au changement de couleur d'affichage du précédent symbole, sont sautées (il n'y a pas de précédent symbole au début du déroulement du programme).

On verra (étiquette 370) que K est posé égel au numéro de code ASCII R de la touche freppée. Per suite K n'a bien la veleur -1 qu'en début de programme.

*** 340 - 360

Ces instructions sont destinées à chenger la couleur du précédent symbole affiché. Nous les étudierons après avoir exeminé les instructions d'effichage du symbole correspondant à la dernière touche freppée.

*** 370

a/ K est posé égal à R, numéro de code ASCII de la dernière touche frappée.

b/ Les numéro de code ASCII de 27 à 32 ne sont pas des numéros de code de carectères "controle". Ils ne donnent donc pas lieu à l'affichage d'un cerectère alphabétique correspondant. Par contre trois d'entre eux correspondent à une touche de fonction (28 INS, 29 EFF, 30 Flèche Retour) et un quatrième, 32, correspond à le barre d'espace.

Par suite si K est compris entre 26 et 33 (bornes exclues), le programme saute l'étiquette 380 contenant les instructions d'affichage des ceractères elphanumériques à l'écren.

*** 380

a/ Positionne le curseur en colonne C(K), ligne L(K) correspondant au symbole numéro K.

b/ Fixe la couleur des caractères au Rouge (1), sur fond Cyen (6).

c/ Affiche le caractère RN$

d/ Pour pouvoir changer la couleur du caractère qui vient d'être affiché, dès qu'une nouvelle touche est frappée, il est nécessaire de garder en mémoire le symbole et son numéro (c'est K qui n'est posé égal à R, en 370, qu'une fois le changement de couleur effectué). Aussi, une fois affiché RN$, pose-t-on RA$ = RN$ (A pour "ancien" et N pour "nouveau"). C'est donc le symbole RA$ de numéro K dont on modifiera la couleur (en 340 - 360), alors que l'on a déjà enregistré le numéro R du nouveau symbole RN$.

Nous pouvons maintenant étudier les instructions 340 - 360 de changement de couleur.

*** 340

Si le caractère est un caractère "controle" (numéro de code ASCII inférieur à 27), il doit être affiché en Blanc (7) sur fond Magenta (5), sinon il est affiché en Bleu (4) sur fond Cyan (6). Les deux paramètres C1 et C2 sont alors posés égaux respectivement à 7 et 5 dans le premier cas sinon égaux à 4 et 6 dans le second cas. C1 et C2 seront utilisés dans l'instruction COLOR en 360.

*** 350

Si K est dans l'intervalle 26 - 33, bornes exclues, aucun symbole alphabétique n'a été affiché (voir étiquette 370). Il est donc nécessaire de sauter l'étiquette 360 de changement de couleur du dernier symbole.

*** 360

a/ Place le curseur en colonne C(K), ligne L(K) correspondant au symbole de numéro K (le dernier affiché).

b/ Fixe la couleur caractère à C1 et la couleur fond à C2, C1 et C2 ayant été déterminés en 340.

c/ Affiche le symbole RA$ avec les couleurs C1, C2.

Reprenons maintenant la suite des instructions, là où nous les avions laissées pour retourner aux instructions de changement de couleur.

*** 390

Nous avons affiché les ceractères alphanumériques sans tenir compte jusqu'ici des touches de fonction, c'est à dire que la frappe d'une touche de fonction a eu pour seul effet l'affichage du caractère "controle" correspondant s'il existait.

Il nous feut donc maintenant vérifier que le numéro de code ASCII R du dernier carectère frappé correspond ou non à une touche de fonction (codes 2, 8, 9, 10, 11, 12, 13, 22, 28, 29, 30 et 32).

La varieble, figurant comme paramètre dans l'instruction ON GOTO est celculée à pertir de variables logiques de la forme (K = I), I prenant les diverses valeurs de K correspondant aux touches de fonction. Elle prend les valeurs 2, 3,...,12, 13 suivant que K est égal à 2, 8,...,30, 32 et la valeur 1 si K n'est égal à aucun des codes de touche de fonction.

L'instruction ON GOTO branche alors à la première étiquette, soit 310 pour K quelconque (la touche n'est pas une touche de fonction, auquel cas il suffit de se brancher en 310 pour attendre une nouvelle frappe). Elle branche à le deuxième étiquette, soit 400, pour K = 2 (touche STOP), à la troisième, soit 500, pour K = 8 (touche Flèche Gauche),...,à la treizième étiquette, soit 1500, pour K = 32 (Barre d'espece).

400 - 1510
Toutes ces instructions ont la même structure.

Elles sont destinées à afficher les touches de fonction. Elles sont branchées par l'instruction 390.

Nous n'examinerons que la première d'entre elles en nous bornant à ne signaler que les quelques cas particuliers.

400
Cette étiquette est atteinte pour K = 2 (numéro de code ASCII de la touche STOP)

a/ Place le curseur en colonne 1, ligne 3.

b/ Fixe la couleur caractère au Rouge (1) sur fond Cyan (6).

c/ Affiche la chaine "STP" pour la touche STOP.

410
Le changement de couleur de la chaine de caractères affichée en Rouge par l'instruction précédente ne peut se faire par la méthode utilisée en 340 - 360.

Il est donc nécessaire de prévoir une instruction particulière qui dès que la touche suivante a été frappée effectue le changement de couleur.

a/ Met en attente le programme jusqu'à ce qu'une touche ait été frappée. Cette instruction joue le même rôle que l'instruction en 310.

b/ Place le curseur en colonne 1, ligne 3, c'est à dire sur la première lettre de la chaine STP, précédemment affichée en Rouge.

c/ Fixe la couleur des caractères au Noir (0) sur fond Cyan (6).

d/ Affiche la chaine STP en Noir.

e/ La dernière touche frappée ayant été mise en mémoire dans la variable R$, le programme est branché en 320 pour reprendre le cours du programme principal.

* 1400 - 1410
La seule différence avec les autres étiquettes réside dans l'instruction ATTRB 1,0 qui permet de tracer le caractère utilisateur représentant la Flèche Retour en double largeur. On notera que l'étiquette 1410 comporte une instruction ATTRB 0,0 pour ramener la dimension des caractères à la valeur normale.

* 1500 - 1510
La première instruction passe au Rouge fond (-2) l'intérieur du cadre dessiné en 100 pour représenter la barre d'espace et la seconde, après recueil de la touche suivante passe l'intérieur du cadre au Magenta (-6).

En définitive le programme est le suivant:

```
10 CLS:SCREEN 0,6,6:CLEAR ,,5:LOCATE 0,0,0

20 Y0=14:DY=26:BOX (4,Y0)-(33,Y0+DY):
   BOX (36,Y0)-(260,Y0+DY):
   BOX (268,Y0)-(284,Y0+DY):
   BOX (290,Y0)-(307,Y0+DY)
30 S=12:FOR I=4 TO 15:GOSUB 2000:NEXT
```

```
40 Y0=54:BOX (44,Y0)-(236,Y0+DY):
   BOX (268,Y0)-(284,Y0+DY):
   BOX (290,Y0)-(307,Y0+DY)

50 S=4:FOR I=4 TO 14:GOSUB 2000:NEXT

60 Y0=94:BOX (4,Y0)-(33,Y0+DY):
   BOX (52,Y0)-(212,Y0+DY):
   BOX (215,Y0)-(264,Y0+DY):
   BOX (268,Y0)-(307,Y0+DY)

70 S=12:FOR I=5 TO 13:GOSUB 2000:NEXT

80 Y0=134:BOX (60,Y0)-(204,Y0+DY):
   BOX (268,Y0)-(286,Y0+DY):
   BOX (288,Y0)-(307,Y0+DY)

90 S=4:FOR I=5 TO 13:GOSUB 2000:NEXT

100 BOX (71,174)-(200,199)


150 DEFGR$(0) = 32,64,255,65,33,1,1,255:
    DEFGR$(1) = 156,144,144,152,144,144,144,156:
    DEFGR$(2) = 231,132,132,198,132,132,132,132:
    DEFGR$(3) = 233,73,73,77,75,73,73,233:
    DEFGR$(4) = 58,66,66,34,18,10,10,114

200 DATA 5,2,7,2,9,2,11,2,13,2,15,2,17,2,19,2,
         21,2,28,9,27,4,20,19,25,4,22,19,26,9

210 DATA 23,4,5,4,7,4,9,4,11,4,13,4,15,4,17,4,
         19,4,21,4

220 DATA 28,7,27,2,20,17,25,2,22,17,26,7,24,19

230 DATA 6,9,16,19,12,19,11,14,10,9,13,14,15,
         14,17,14,20,9,19,14,21,14,23,14

240 DATA 25,14,18,19,22,9,24,9,7,14,12,9,9,14,
         14,9,18,9,14,19,8,19,10,19,16,9,8,9,
         0,0,0,0,0,0,24,17
```

```
250 DIM C(94),L(94)

260 FOR I=33 TO 94:READ C(I),L(I):NEXT

270 RESTORE 230:FOR I=1 TO 26:
    READ C(I),L:L(I)=L-2:NEXT

280 A$=CHR$(22):FG$=A$+",":FD$=A$+".":
    FH$=A$+"-":FB$=A$+"/"


300 COLOR 4:K=-1:LOCATE 5,0,0:
    PRINT "TESTEZ LES DIFFERENTES TOUCHES"

310 R$=INPUT$(1)

320 R=ASC(R$):IF R)96 THEN CLS:SCREEN 4,6,6:END
    ELSE IF R(27 THEN RN$=CHR$(R+64) ELSE RN$=R$

330 IF K=-1 GOTO 370

340 IF K(27 THEN C1=7:C2=5 ELSE C1=4:C2=6

350 IF K)26 AND K(33 GOTO 370

360 LOCATE C(K),L(K):COLOR C1,C2:PRINT RA$

370 K=R:IF K)26 AND K(33 GOTO 390

380 LOCATE C(K),L(K):COLOR 1,6:
    PRINT RN$:RA$=RN$

390 ON 1-(K=2)-2*(K=8)-3*(K=9)-4*(K=10)
    -5*(K=11)-6*(K=12)-7*(K=13)-8*(K=22)
    -9*(K=28)-10*(K=29)-11*(K=30)-12*(K=32)
    GOTO 310,400,500,600,700,800,900,
    1000,1100,1200,1300,1400,1500

400 LOCATE 1,3:COLOR 1,6:PRINT "STP"

410 R$=INPUT$(1):LOCATE 1,3:
    COLOR 0,6:PRINT "STP":GOTO 320

500 LOCATE 34,8:COLOR 1,6:PRINT FG$

510 R$=INPUT$(1):LOCATE 34,8:
    COLOR 0,6:PRINT FG$:GOTO 320
```

```
600 LOCATE 37,8:COLOR 1,6:PRINT FD$

610 R$=INPUT$(1):LOCATE 37,8:
    COLOR 0,6:PRINT FD$:GOTO 320

700 LOCATE 37,3:COLOR 1,6:PRINT FB$

710 R$=INPUT$(1):LOCATE 37,3:
    COLOR 0,6:PRINT FD$:GOTO 320

800 LOCATE 34,3:COLOR 1,6:PRINT FH$

810 R$=INPUT$(1):LOCATE 34,3:
    COLOR 0,6:PRINT FH$:GOTO 320

900 LOCATE 1,13:COLOR 1,6:PRINT "RAZ"

910 R$=INPUT$(1):LOCATE 1,13:
    COLOR 0,6:PRINT "RAZ":GOTO 320

1000 LOCATE 27,13:COLOR 1,6:
     PRINT "ENTREE"

1010 R$=INPUT$(1):LOCATE 27,13:
     COLOR 0,6:PRINT "ENTREE":GOTO 320

1100 LOCATE 29,3:COLOR 1,6:PRINT "ACC"

1110 R$=INPUT$(1):LOCATE 29,3:
     COLOR 0,6:PRINT "ACC":GOTO 320

1200 LOCATE 34,18:COLOR 1,6:
     PRINT GR$(3);GR$(4)

1210 R$=INPUT$(1):LOCATE 34,18:
     COLOR 0,6:PRINT GR$(3);GR$(4):GOTO 320

1300 LOCATE 36,18:COLOR 1,6:
     PRINT GR$(1);GR$(2)

1310 R$=INPUT$(1):LOCATE 36,18:
     COLOR 0,6:PRINT GR$(1);GR$(2):GOTO 320
```

```
1400 LOCATE 35,13:COLOR 1,6:
     ATTRB 1,0:PRINT GR$(0)

1410 R$=INPUT$(1):LOCATE 35,13:
     COLOR 0,6:PRINT GR$(0):ATTRB 0,0:
     GOTO 320


1500 BOXF(72,175)-(199,198),-2

1510 R$=INPUT$(1):
     BOXF(72,175)-(199,198),-6:GOTO 320


2000 X=16*I-S:LINE (X,Y0)-(X,Y0+DY):RETURN
```

7

Miniécriture

Le BASIC TO7 dispose d'un jeu de caractères s'inscrivent dans un carré de 8 x 8 points. Il est par ailleurs possible, grâce à l'instruction ATTRB -(M.Ref 26)- de tracer ces mêmes caractères dans des rectangles 8 x 16, 16 x 8 ou des carrés 16 x 16.

Par contre il n'est pas possible de tracer des caractères plus petits. Or dans certaines applications, lorsque l'écran est très encombré, il pourrait être fort utile de disposer de caractères plus petits.

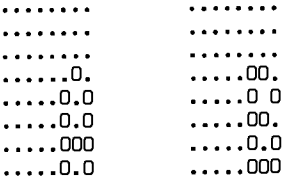
A première vue le problème semble insoluble, puisque les caractères dits utilisateur, définis par l'instruction DEFGR$(N) et affichés par PRINT GR$(N), occupent eux aussi un carré de 8 x 8 points.

Il nous faudrait donc bâtir des caractères utilisateur regroupant deux minicaractères dans un même carré 8 x 8. En admettant que l'on se contente de reproduire seulement les 26 majuscules, il nous faudrait bâtir 26 x 26 = 676 caractères représentant toutes les combinaisons de deux lettres. Outre le travail fastidieux de définition de ces 676 caractères, on tombe sur une impossibilité du fait que l'on ne peut stocker plus de 128 caractères utilisateur !

Les quelques remarques suivantes, reposant sur la façon dont sont constitués les caractères utilisateur, vont nous permettre de trouver une solution.

Bâtissons le caractère A inscrit dans un rectangle de trois points de large et de cinq points de haut et que nous plaçons dans le coin bas-droite de la matrice 8 x 8 des caractères utilisateur.

Bâtissons de même un second caractère, B par exemple avec les mêmes règles.



Le caractère A est représenté par les huit nombres binaires suivants avec leurs équivalents décimaux :

```
0 0 0 0 0 0 0 0  =  0
0 0 0 0 0 0 0 0  =  0
0 0 0 0 0 0 0 0  =  0
0 0 0 0 0 1 0 0  =  2
0 0 0 0 1 0 1 0  =  5
0 0 0 0 1 0 1 0  =  5
0 0 0 0 1 1 1 0  =  7
0 0 0 0 1 0 1 0  =  5
```

Prenons la dernière ligne 0 0 0 0 1 0 1 0 (= 5).

Si nous multiplions par deux sa valeur décimale, nous obtenons 0 0 0 1 0 1 0 0. Si nous multiplions par 4 sa valeur décimale, nous obtenons 0 0 1 0 1 0 0 0. Chaque multiplication par 2 de la valeur décimale se traduit donc par un déplacement de la ligne de points correspondante de un point vers la gauche.

Ainsi en multipliant par 32, chacune des huit valeurs décimales de définition du mini A, placé en bas à droite du carré 8 x 8, nous obtenons les huit valeurs caractéristiques d'un caractère utilisateur représentant un mini A, placé en bas à gauche du même carré.

Les valeurs caractéristiques du mini A placé à gauche sont donc 0, 0, 0, 64, 160, 160, 224, 160. Si maintenant nous voulons placer le mini A en haut à gauche, il nous suffit de permuter les valeurs des huit lignes de la façon suivante : 64, 160, 160, 224, 160, 0, 0, 0.

Replaçons notre mini A en bas à gauche et considérons la ligne inférieure caractérisée par 1 0 1 0 0 0 0 0 = 160.

Si nous ajoutons à cette valeur décimale, la valeur décimale correspondant à la dernière ligne du mini B placé en bas à droite (0 0 0 0 0 1 1 1 = 7), nous obtenons 1 0 1 0 0 1 1 1 = 167.

Ainsi en ajoutant aux valeurs décimales du mini A en bas à gauche les valeurs décimales du mini B à droite, nous obtenons les valeurs décimales du caractère composé du mini A à gauche et du mini B à droite.

Nous sommes maintenant en mesure de "calculer" tout caractère utilisateur composé de deux mini caractères dont on connait les cinq valeurs caractéristiques (pour les situer en "hauteur" à l'intérieur du carré 8 x 8 il suffit de compléter ces cinq valeurs par trois zéros répartis entre le début et la fin de liste (3 zéros en début de liste pour placer les deux caractères sur la ligne du bas, 3 zéros en fin de liste pour les placer "collés" en haut du carré, etc).

En pratique nous disposerons nos deux caractères en laissant libre la première colonne de points du carré 8 x 8. Ainsi le caractère AB sera :

```
........
........
........
..0..00.
.0.0.0.0
.0.0.00.
.000.0.0
.0.0.000
```

Si NA(L) est la valeur décimale caractérisant la ligne L du mini A (placé à droite) et si NB(L) est la valeur décimale caractérisant la ligne L du mini B (placé à droite également), la valeur décimale de la ligne L du caractère AB est donc donnée par

$$NAB(L) = 16 * NA(L) + NB(L)$$

Nous n'avons multiplié NA(L) que par 16, puisque nous ne déplaçons le caractère que de 4 points vers la gauche, pour laisser vide la première colonne du carré.

Pour obtenir toutes les combinaisons de minicaractères, il suffit donc de stocker les 26 x 5 valeurs décimales correspondant à chacun des 26 caractères alphabétiques A à Z et de calculer successivement, en regroupant deux à deux les caractères de la chaîne à écrire, chacun des caractères utilisateur correspondants. Oès qu'un caractère calculé est affiché par un PRINT il est remplacé par le nouveau caractère calculé. Il suffit donc de réserver une seule place mémoire pour le caractère GR$(0), qui sera redéfini par OEFGR$(0) autant de fois que nécessaire.

Pourquoi alors s'en tenir aux seules 26 majuscules ? Il est possible de mettre en mémoire les 63 groupes de 5 valeurs décimales correspondant aux caractères de numéro de code ASCII variant de 32 à 94 (il ne faut pas oublier l'espace, de numéro de code ASCII 32, dont les 5 valeurs sont évidemment nulles).

Nous allons donc d'une part bâtir un premier programme pour stocker les 315 valeurs caractérisant nos 63 caractères puis un deuxième programme d'utilisation permettant d'écrire une chaîne de minicaractères. Ce deuxième programme sera d'ailleurs écrit sous forme de sous-programme, car il ne doit constituer qu'un élément d'un programme principal de "dessin" sur l'écran.

* * * * * *

Le programme de stockage des données a pour but de placer les 315 valeurs en mémoire de façon à les récupérer aisément pour "calculer" chacun des caractères utilisateur.

Chaque minicaractère élémentaire est caractérisé par 5 valeurs comprises entre 0 et 7, puisque les points de chaque ligne ne peuvent occuper que les trois places de droite (aucun point 0 0 0 0 0 0 0 0 = 0, les trois points 0 0 0 0 0 1 1 1 = 7)

Il est donc commode de les stocker directement dans les mémoires disponibles, par des instructions POKE -(M.Ref 86)-, pour les récupérer ultérieurement par des instructions PEEK -(M.Ref 132)-.

Le problème que l'on se pose généralement est celui de la place mémoire la plus favorable pour effectuer ce stockage.

Il est nécessaire pour répondre de connaître comment est organisée la mémoire. L'annexe F du Manuel de Référence -(M.Ref 221)- donne quelques renseignement à ce sujet.

Les mémoires ont une adresse variant de 0 à 65535 (en numérotation héxadécimale $0 à $FFFF le signe $ signifiant que la valeur numérique qui suit est exprimée en héxadécimal). Une note en fin de ce chapitre donne la façon pratique pour passer de la numérotation décimale à la numérotation héxadécimale en utilisant le TO7.

Les 16383 premières mémoires ($0 à $3FFF) sont réservées au BASIC et résident physiquement dans la cartouche "mémo 7" enfichable.

Les mémoires de 16384 à 24575 ($4000 à $5FFF) constituent la mémoire écran. C'est dans ces octets que sont stockées les données forme et couleur.

Les mémoires de 24576 à 24831 ($6000 à $60FF) sont utilisées par le système pour y loger ses propres données. Ces 256 octets portent le nom de Page 0.

Toutes ces mémoires sont bien entendu inutilisables pour y stocker des données.

Les mémoires de 24832 à 61439 ($6100 à $0FFF) sont disponibles pour l'utilisateur.

En pratique, si l'on n'a pas installé de mémoire additionnelle, seules les 7936 premières mémoires de 24832 à 32767 ($6100 à $7FFF) sont utilisables. On dit que la mémoire disponible est de 8K. Ces 8 Kilooctets, c'est à dire 8 x 1024 = 8192 constituent les mémoires comptées à partir de 24576 ($6000). En toute rigueur il y a un peu moins de 8K disponibles puisque les 256 octets de la Page 0 ne sont pas utilisables.

Par contre avec les 16K de mémoire additionnelle, il y a 24K = 24 x 1024 = 24576 mémoires de 24576 à 49151 ($6000 à $BFFF) dont 24320 sont disponibles de 24832 à 49151 ($6100 à $BFFF).

Enfin pour en terminer avec la mémoire, sachons que la plage 61440 à 65535 ($E000 à $FFFF) est attribuée au système et est bien entendue inutilisable pour y stocker des données.

On sait que l'instruction CLEAR I,J,K -(M.Ref 32)- permet de réserver de la place mémoire pour y stocker des données ou des programmes en langage machine. Le paramètre J fixe la plus haute adresse mémoire que l'interpréteur BASIC peut utiliser pour stocker le programme et les variables. Les mémoires au dessus de J sont donc protégées et il est possible d'y stocker des programmes en langage machine ou des données.

En conséquence, si nous disposons d'une mémoire additionnelle, nous logerons nos 315 valeurs à partir de 48831, en posant le paramètre J de CLEAR égal à 48830 (cela laisse quelques mémoires disponibles aux adresses les plus hautes, mais nous n'en sommes pas à quelques octets près).

Par contre si nous n'avons pas de mémoire additionnelle nous logerons nos 315 valeurs à partir de 32451, en posant le paramètre J égal à 32450.

Une fois ces valeurs stockées en mémoire il est indispensable de les sauver sur la cassette pour pouvoir les récupérer ultérieurement pour le programme d'utilisation.

Cette sauvegarde se fait par l'instruction SAVEM -(M.Ref 181)-. Nous en verrons le détail dans le programme lui-même.

Le programme se présente alors de la façon suivante :

```
10 CLS:SCREEN 4,6,6:CLEAR ,32450:M0=32451

20 DATA 0,0,0,0,0,2,2,2,0,2,5,5,0,0,0,0,0,7,5,7,
        3,6,2,3,6

21 DATA 4,1,2,4,1,3,3,6,7,2,1,2,0,0,0,2,4,4,4,2,
        2,1,1,1,2

22 DATA 0,0,5,2,5,0,0,2,7,2,0,0,0,2,4,0,0,0,7,0,
        0,0,0,0,2
```

```
23 DATA 0,0,1,2,4,7,5,5,5,7,2,6,2,2,7,7,1,3,4,7,
        7,1,3,1,7

24 DATA 1,3,5,7,1,7,4,7,1,7,2,4;7,5,7,7,1,1,1,1,
        7,5,7,5,7

25 DATA 7,5,7,1,7,0,0,2,0,2,0,2,0,2,4,1,2,4,2,1,
        0,0,7,0,7

26 DATA 4,2,1,2,4,7,1,2,0,2,7,5,1,7,7,2,5,5,7,5,
        6,5,6,5;7

27 DATA 7,4,4,4,7,6,5,5,5,6,7,4,6,4,7,7,4,6,4,4,
        7,4,5,5,7

28 DATA 5,5,7,5,5,7,2,2,2,7,1,1,1,5,7,5,6,6,5,5,
        4,4,4,4,7

29 DATA 5,7,5,5,5,5,7,7,7,5,7,5,5,5,7,7,5,7,4,4,
        7,5,5,7,7

30 DATA 7,5  ',6,5,7,4,7,1,7,7,2,2,2,2,5,5,5,5,7,
        5,5,5,5,2

31 DATA 5,5,5,7,5,5,5,2,5,5,5,5,2,2,2,7,1,2,4,7,
        6,4,4,4,6

32 DATA 0,0,4,2,1,3,1,1,1,3,2,7,2,2,2

200 FOR I=0 TO 314

210 READ Z:POKE M0+I,Z

220 LOCATE 15,12:PRINT M0+I;Z

230 NEXT

300 SAVEM "DATAMIN$",M0,32767,0
```

*** 10

a/ Efface l'écran.

b/ Fixe la couleur des caractères au Bleu (4), du fond et de l'entourage au Cyan (6).

c/ Met à zéro toutes les variables. Réserve la plage mémoire de l'adresse 32450 jusqu'à l'adresse maximale correspondant à la configuration de la machine.

Nous avons supposé ici que la machine ne dispose pas de mémoire additionnelle. L'adresse maximale est donc 32767. On a ainsi réservé la place d'un peu plus de 315 octets.

Dans le cas où l'on dispose de la mémoire additionnelle de 16K, l'instruction CLEAR doit s'écrire CLEAR ,48830 car l'adresse maximale est dans ce cas 49151 (en toute rigueur on aurait pu retenir CLEAR ,48836 pour loger exactement nos 315 octets).

On notera la virgule précédant la valeur numérique, précisant que cette valeur fixe le second paramètre, le premier étant pris égal à 300 par défaut pour le traitement des chaînes de caractères. On notera en outre qu'il n'y a pas de virgule après la valeur numérique, le dernier paramètre étant dans ce cas pris par défaut égal à zéro (nombre de caractères utilisateur, nul dans ce programme, ce qui ne sera pas le cas dans le programme principal).

d/ M0 est l'adresse de la première mémoire de stockage des données (32451 sans mémoire additionnelle, 48831 avec mémoire additionnelle).

*** 20 - 32
Les données sont tout d'abord mémorisées sous forme de DATA -(M.Ref 38)- car il est plus commode de les taper et de les vérifier sous cette forme.

Les données sont placées par groupes de 5 x 5 sous une même étiquette, chaque paquet de cinq correspondant aux cinq paramètres caractéristiques d'un caractère. Les caractères ont été bâtis, par le programme de conception du Chapitre V -(M.Bas 63)- dans l'ordre des numéro de code ASCII à partir du numéro de code 32 (espace) jusqu'au numéro de code 94.

Cette disposition des DATA permet une vérification partielle sur le nombre de données figurant sous une étiquette. Les derniers chiffres doivent s'aligner sur une même colonne.

Une telle précaution peut paraître futile.

On se souviendra toutefois qu'il est très facile de sauter un caractère au cours de la lecture et de la frappe de 315 chiffres ou de taper l'un d'entre eux deux fois. Or une telle erreur conduit à un décalage de toutes les données provoquant une véritable catastrophe au niveau des futurs caractères en mêlant les paramètres d'un caractère avec ceux du suivant et du précédent.

Bien entendu cela n'empèche pas de commettre des erreurs, mais celles-ci seront limitées à des valeurs erronées se traduisant par une déformation du caractère correspondant, facilement identifiable. La reconnaissance du caractère déformé et donc de son numéro de code ASCII permettra alors de retrouver facilement les cinq paramètres correspondants, d'autant plus qu'ils sont regroupés par paquets de cinq.

*** 200
Ouverture d'une boucle en I de 0 à 314

*** 210
a/ Lecture dans les DATA du paramètre Z.

Rappelons que les données -(M.Ref 38)- sont lues dans l'ordre où elles figurent dans les instructions DATA.

b/ La valeur Z est chargée dans l'octet d'adresse MO+I -(M.Ref 86)-.

*** 220
a/ Place le curseur en colonne 15, ligne 12, c'est à dire sensiblement au milieu de l'écran.

b/ Affiche l'adresse de la mémoire que l'on vient de charger et la valeur chargée.

Cette dernière instruction n'est pas très utile pour la vérification des données car les valeurs défilent trop vite pour que l'opérateur puisse les distinguer.

Elle permet toutefois à l'opérateur de "patienter", car il voit défiler les valeurs d'adresse de MO jusqu'à MO+314.

Nous verrons un peu plus loin une véritable méthode de vérification.

*** 230
Fermeture de la boucle en I. Une seule boucle étant en service, il n'est pas nécessaire de préciser le nom de la variable I -(M.Ref 55)-.

*** 300
Les données stockées de l'octet d'adresse ¯?451 à l'octet 32767 (adresse maximale dans la configuration sans mémoire additionnelle), sont sauvegardées sur la cassette -(M.Ref 181)-.

Bien entendu avant de lancer le programme il est nécessaire de mettre le lecteur de cassette en situation d'enregistrement, les deuxième et troisième touches enfoncées.

Le nom du fichier de stockage a été choisi arbitrairement, tout en répondant aux règles fixées pour les noms de fichier -(M.Ref 157)-, mais nous avons pris un nom ayant une signification mnémotechnique (n'appelez pas tous vos fichiers données OATA ou OONNEES sous peine de ne plus être capable de les reconnaître ultérieurement). Comme nous n'avons pas indiqué de suffixe derrière un point, ce fichier sera reconnu à la relecture de la cassette par "OATAMIN$.BIN", le suffixe BIN placé par l'interpréteur précisant que c'est un fichier binaire.

On notera que la première adresse figurant après une virgule est l'adresse basse, la seconde l'adresse haute. Le dernier 0 figurant après la seconde virgule doit impérativement figurer. Il représente l'adresse d'appel du sous programme ainsi sauvegardé en langage machine. Dans notre cas où les octets sauvegardés sont des données et non un programme, l'adresse d'exécution du sous programme n'a pas de sens mais il est nécessaire de faire figurer une valeur sous peine d'avoir un message d'erreur SN (Syntax Error).

* * * * * *

Etudions maintenant une modification du programme précédent permettant de vérifier les données. Bien entendu une première méthode consiste à relire soigneusement l'ensemble des données, de préférence avec l'aide d'un second opérateur dictant les valeurs à vérifier.

Une telle méthode est très fastidieuse et encore nous n'avons à vérifier que 315 valeurs ! Par ailleurs rien n'empèche des erreurs de transcription entre le programme de conception et le listing des valeurs, surtout si, ne disposant pas d'imprimante, nous avons relevé les valeurs à l'écran.

Or il est très facile de constater qu'un caractère est "tordu", beaucoup plus que de constater que l'on a tapé un 7 au lieu d'un 1.

Nous allons donc afficher à l'écran chaque caractère dès que cinq valeurs consécutives ont été lues dans les DATA. Il suffit pour cela de bâtir le caractère par DEFGR$(0). Voyons le programme ainsi modifié entre 200 et 300.

```
200 J=0:K=31

210 FOR I=0 TO 314

220 READ Z:POKE M0+I,Z

230 Z(J)=Z

240 J=J+1:IF J=5 THEN J=0:K=K+1:
    DEFGR$(0)=0,0,Z(0),Z(1),Z(2),Z(3),Z(4),0:
    LOCATE 15,12:PRINT GR$(0):R$=INPUT$(1)

250 NEXT
```

*** 200

a/ J est un compteur que l'on fera varier de 0 à 4 pour compter les valeurs par paquets de 5. Il est ici initialisé à zéro.

b/ K est un second compteur que l'on fera varier de 32 à 94. Il est ici initialisé à 31.

*** 210

Ouverture de la boucle en I comme dans le programme précédent.

*** 220

Lecture de Z dans les DATA et copie dans l'octet d'adresse M0+I.

*** 230
La valeur Z est stockée provisoirement dans la variable Z(J). Cela permettra quand le compteur J aura atteint la valeur 4 de disposer des cinq valeurs caractéristiques du caractère. On notera que Z(J) n'a pas été dimensionnée par OIM -(M.Ref 46)- puisque J ne dépasse pas 10.

*** 240
a/ J est incrémenté de 1.

b/ si J après incrémentation atteint la valeur 5, il est remis à zéro. Dans ce cas on a recueilli les cinq valeurs du caractère suivant.

On peut alors incrémenter le compteur K de 1 (c'est pourquoi la valeur initiale de K est 31 de façon à obtenir pour le premier caractère la valeur 32, qui est le numéro de code ASCII de l'espace). On définit ensuite le caractère utilisateur par DEFGR$(0) en encadrant les cinq valeurs Z(0) à Z(4) par deux zéros à gauche et un zéro à droite. Le caractère ainsi bâti se situera en conséquence sur les lignes 2 à 6 du carré 8 x 8, la ligne 1 étant la ligne inférieure.

Le caractère GR$(0) est enfin affiché en colonne 15, ligne 12.

Pour ne pas voir défiler les caractères trop rapidement sans pouvoir les controler, on dispose une instruction d'arret R$=INPUT$(1) qui stoppe le déroulement du programme tant qu'une touche quelconque n'a pas été frappée.

On ne place pas de message à l'opérateur, ni de test sur la touche comme dans de nombreux programmes précédents car il s'agit là d'un programme de mise au point et non d'un programme opérationnel destiné à être mis entre les mains d'opérateurs non avertis.

*** 250
Fermeture de la boucle en I.

Bien entendu on prendra garde à ne pas oublier de modifier l'instruction CLEAR à l'étiquette 10 pour réserver une place mémoire pour le caractère utilisateur GR$(0), par CLEAR ,32450,1.

Une fois sauvée notre liste de données, il nous reste à bâtir le programme d'écriture lui-même.

Nous allons toutefois commencer par un programme destiné à vérifier qu'il est possible de récupérer les données sur la cassette. Il s'inspire du programme de test précédent.

```
10 CLS:SCREEN 4,6,6:CLEAR ,32450,1:M0=32451

20 FOR I=0 TO 314:POKE M0+I,0:NEXT

30 LOADM "DATAMIN$"

200 J=0:K=31

210 FOR I=0 TO 314

220 Z=PEEK(M0+I)

230 Z(J)=Z

240 J=J+1:IF J=5 THEN J=0:K=K+1:
    DEFGR$(0)=0,0,Z(0),Z(1),Z(2),Z(3),Z(4),0:
    LOCATE 15,12:PRINT GR$(0):R$=INPUT$(1)

250 NEXT
```

Ce programme test n'a pas été optimisé. Il est évident que les instructions 220 et 230 pourraient être regroupées en une seule. Rappelons qu'il ne s'agit là que d'un programme de mise au point et de vérification. Il est plus commode de le bâtir en modifiant quelques instructions du programme précédent. Il est inutile de perdre son temps à optimiser un tel programme.

L'instruction 20 est destinée à effacer de la mémoire les données chargées par le précédent programme.

Il faut prendre garde en effet à ce qu'une non relecture des données pourrait être masquée, si nous laissions en mémoire les données précédentes. Il ne faut pas oublier en effet que, contrairement aux variables, les mémoires protégées par l'instruction CLEAR ne sont pas mises à zéro par la commande RUN. Seule une instruction analogue à l'instruction 20 peut remettre

ces mémoires à zéro (ou encore la coupure de l'interrupteur général du TO7, ce que nous n'avons pas fait pour pouvoir disposer du programme précédent et le modifier).

L'instruction 30 charge le fichier "OATAMIN$".

On notera que l'instruction de chargement d'un fichier binaire n'est pas LOAD, comme pour un fichier BASIC, mais LOADM. Un LOAD conduirait à une erreur FM (Bad File Mode, Mode d'accès au fichier incorrect).

Les instructions suivantes sont directement inspirées du programme précédent. Il est inutile de les commenter.

Pour bien vous convaincre que les mémoires de stockage des données ne sont pas effacées par la commande RUN, vous pouvez tenter l'expérience suivante dans la mesure où vous avez effectivement vérifié la relecture des données.

Supprimez les instructions 20 et 30 d'effacement et de lecture du fichier et relancez le programme. Vous pourrez constater qu'il fonctionne correctement. (N'oubliez pas de supprimer 20 sous peine de déboires!)

* * * * * *

Nous en arrivons enfin au programme d'écriture proprement dit.

Nous allons bâtir un programme principal simulant un programme opérationnel de dessin en demandant simplement à l'opérateur de préciser la chaîne de caractères qu'il veut placer sur l'écran ainsi que la colonne et la ligne où sera situé le premier caractère.

Nous bâtirons ensuite un sous-programme d'écriture de cette chaîne en minicaractères. C'est bien évidemment le sous-programme qui sera sauvegardé pour être ajouté à de vrais programmes de dessin.

```
10 CLS:SCREEN 4,6,6

20 CLEAR ,32450,1:M0=32451:MA=M0-160

30 LOADM "DATAMIN$"


500 LOCATE 0,0:PRINT "C =":
    LOCATE 20,0:PRINT "L ="

510 LOCATE 0,2:PRINT "Z =":
    LOCATE 20,2:PRINT "Z$ ="

520 LOCATE 3,0:INPUT CC:LOCATE 23,0:INPUT LL

530 LOCATE 3,2:INPUT ZL:LOCATE 23,2:INPUT Z$

540 CLS:GOSUB 60000

550 R$=INPUT$(1):CLS:IF R$="F" THEN END
    ELSE GOTO 500


60000 ATTRB 0,0:L=LEN(Z$):
      IF L MOD 2 = 1 THEN L=L+1:Z$=Z$+" "

60010 FOR ZZ=0 TO 7:U(ZZ)=0:NEXT ZZ

60020 FOR Z=1 TO L STEP 2

60030 M1=MA+5*ASC(MID$(Z$,Z,1)):
      M2=MA+5*ASC(MID$(Z$,Z+1,1))

60040 FOR ZZ=0 TO 4:
      U(ZZ+ZL)=16*PEEK(M1+ZZ)+PEEK(M2+ZZ):
      NEXT ZZ

60050 DEFGR$(0)=U(0),U(1),U(2),U(3),
                U(4),U(5),U(6),U(7)

60060 LOCATE CC,LL:PRINT GR$(0);:CC=CC+1

60070 NEXT Z

60080 RETURN
```

*** 10

a/ Efface l'écran

b/ Fixe les couleurs générales écran

*** 20

a/ Réservation des mémoires dont les adresses sont supérieures à 32450, pour y loger les données caractère et réservation d'une place pour un caractère utilisateur. Dans le cas où l'on dispose d'une mémoire additionnelle, il faut déclarer une adresse de protection égale à 48830.

b/ M0 est la première mémoire de stockage des données.

c/ Nous verrons qu'il est utile de disposer du paramètre MA pour le calcul, dans le sous-programme, des adresses de stockage des données correspondant à un caractère.

*** 30

Chargement en mémoire centrale du fichier binaire contenant les données caractère.

Les deux instructions 20 et 30 doivent toujours figurer dans le programme principal.

*** 500 - 530

Ces instructions permettent d'entrer la chaîne de caractères Z$, le numéro de colonne CC, le numéro de ligne LL où sera écrit le premier caractère.

Nous disposons en outre d'un dernier paramètre de "réglage" de position de la chaîne. C'est sa hauteur ZL dans les huit lignes écran attribuées au caractère (qui n'en occupe que cinq), hauteur réglée en disposant trois zéros de part et d'autre des cinq valeurs caractéristiques dans l'instruction OEFGR$(0).

*** 540

Efface l'écran et branche au sous-programme 60000 d'écriture des minicaractères.

Nous avons placé le sous-programme en 60000 de façon à faciliter sa fusion avec un programme principal.

*** 550

a/ Arrête le programme jusqu'à ce qu'une touche ait été frappée.

Là encore nous ne diposons pas de message à l'écran pour prévenir l'opérateur puisqu'il s'agit seulement d'un programme de mise au point et d'essai du sous-programme de miniécriture.

b/ Efface l'écran

c/ Si la touche frappée est la touche "F" le programme est définitivement arrêtá (END), sinon (ELSE) le programme est branché en 500 pour un nouvel essai.

*** 60000

a/ Impose la taille normale des caractères.

Cette instruction est une instruction de précaution au cas où le programme principal aurait effectué le branchement au sous-programme sans remettre les caractères à la taille normale. Il est inutile en effet de tracer des minicaractères en taille double !

b/ L est le nombre de caractères de la chaîne Z$ à écrire.

c/ Si L est impair, le reste de la division de L par deux est égal à 1 (l'opération Modulo N, effectuée par l'instruction MOD N -(M.Ref 19)-, donne le reste de la division par N). Dans ce cas nous ajoutons un caractère "espace" en fin de chaîne de façon à travailler sur une chaîne contenant un nombre pair de caractères et incrémentons L de un.

Les caractères sont en effet tracés par groupes de deux. Il ne faut donc pas laisser un "célibataire" en bout de chaîne.

*** 60010

La variable U(ZZ) est destinée à recevoir les huit valeurs calculées, entrant dans la definition de chacun des caractères utilisateur regroupant deux minicaractères.

Seules cinq valeurs changent d'un caractère à l'autre, les trois autres étant des zéros, la "position" de ces zéros dépendant du paramètre ZL, fixant la "hauteur" de la chaîne dans la ligne LL.

Il est donc nécessaire avant d'écrire la chaîne de remettre les huit valeurs des U(ZZ) à zéro, de façon à ne modifier, d'un caractère à l'autre de la chaîne que les U(ZZ) non nuls.

*** 60020
Ouverture d'une boucle en Z destinée à analyser les caractères de Z$ deux par deux (STEP 2). On voit ici la nécessité de disposer d'une chaîne de longueur paire.

*** 60030
MID$(Z$,Z,1) est le caractère de rang Z de Z$ -(M.Ref 130)-.

De même MID$(Z$,Z+1,1) est le caractère de rang Z+1 de Z$.

La fonction ASC placée devant ces deux expressions fournit donc les numéros de code ASCII des caractères de rangs Z et Z+1.

Les valeurs caractéristiques des minicaractères sont stockées par groupes de cinq, le premier minicaractère étant l'espace, de numéro de code 32. L'adresse de la mémoire de stockage de la première valeur du minicaractère de numéro de code N est donc M0 + 5 * (N - 32) soit M0 -160 + 5 * N. Voilà la raison d'utiliser la variable MA = M0 - 160.

Ainsi M1 et M2 sont les adresses de stockage des premières valeurs caractéristiques des deux caractère de rang Z et Z+1 de la chaîne Z$.

*** 60040
a/ Ouverture d'une boucle en ZZ destinée à calculer les cinq valeurs non nulles du caractère utilisateur regroupant les deux minicaractères de rang Z et Z+1.

b/ ZZ variant de 0 à 4, les U non nuls ont pour indices ZZ+ZL.

Ils sont calculés par addition de la valeur relevée en M2+ZZ pour le second minicaractère (situé à droite du carré 8 x 8) et de la valeur relevée en M1+ZZ, pour le premier minicaractère, préalablement multipliée par 16 (pour le placer à gauche).

c/ Ferme la boucle en ZZ

*** 60050
Définition du caractère utilisateur GR$(0) -(M.Ref 190)-.

*** 60060
a/ Place le curseur en colonne CC, ligne LL.

b/ Affiche le caractère utilisateur GR$(0), précédemment défini.

Le point-virgule n'est utile que dans le cas où l'affichage de la chaîne se ferait sur la dernière ligne écran; il évite une remontée intempestive d'écran à l'affichage de chaque caractère sur cette dernière ligne.

c/ Incrémente CC de un pour déplacer le curseur vers la droite pour le caractère suivant.

*** 60070
Ferme la boucle en Z.

*** 60080
Instruction de clôture du sous-programme -(M.Ref 57)-.

Il pourrait être utile de compléter le sous-programme, en y adjoignant des tests sur la validité des données CC, LL, ZL et Z$.

Il faut que ZL soit compris entre 0 et 3 (bornes incluses), que CC soit compris entre 0 et 39, LL entre 0 et 24. Enfin on peut vérifier que CC + L/2, L étant la longueur de la chaîne est bien inférieur à 40, pour éviter que la chaîne ne soit coupée en deux par un saut à la ligne intempestif. Chacun complètera le sous-programme si nécessaire, en fonction du programme principal d'utilisation.

CONVERSION PRATIQUE HEXADECIMAL - OECIMAL.

A/ Conversion de l'Hexadécimal en Oécimal.

Connaissant une valeur héxadécimale, B32E par exemple, taper directement au clavier :

```
? &HB32E
```

La réponse est donnée en décimal, soit ici 45870.

B/ Conversion du Oécimal en Héxadécimal.

Connaissant la valeur décimale, 45311 par exemple, taper directement au clavier :

```
? HEX$(45311)
```

La réponse est donnée en héxadécimal, soit ici B0FF.

8

Calculette

Nous allons simuler le fonctionnement d'une calculette de poche, munie d'une mémoire et capable d'effectuer les quatre opérations.

Transformer le TO7 en une calculette de poche capable des quatre opérations seulement n'est sûrement pas "rentable", mais les problèmes de programmation posés par cette réalisation sont suffisament nombreux pour que l'exercice soit fructueux.

Nous représenterons la calculette sous forme d'un clavier rectangulaire sur lequel un certain nombre de "touches" seront dessinées, chaque touche étant identifiée par un symbole précisant sa fonction. La commande de la calculette se fera bien entendu en désignant les touches à l'aide du crayon optique.

Il nous faut tout d'abord concevoir le clavier.

Nous devons tracer dix touches pour les dix chiffres plus une onzième pour le point décimal (ou la virgule si vous préférez la notation française). Il y a quatre touches d'opération plus une cinquième pour le signe " = ", permettant d'afficher le résultat de l'opération effectuée. Il faut enfin prévoir deux touches pour la mémoire (mise en mémoire et rappel du contenu de la mémoire), une touche d'annulation de la valeur affichée et une touche de changement de signe.

Nous arrivons ainsi à vingt touches que bien naturellement nous disposerons en quatre rangées de cinq.

Il nous faut également prévoir une fenêtre d'affichage des nombres et, comble de raffinement, une touche "Marche-Arret", bien utile pour retrouver le TO7 pour un usage plus à la mesure de ses capacités.

Enfin à gauche de la fenêtre d'affichage des résultats, située elle même sous le tableau des vingt touches, nous placerons un marqueur identifiant la dernière touche de fonction désignée par le crayon optique.

Nous placerons au centre de chaque touche un symbole en double taille. Ainsi chaque touche sera un rectangle de 6 caractères de large (soit 6 x 8 = 48 points) et 4 caractères de haut (soit 4 x 8 = 32 points) de façon à pouvoir y centrer harmonieusement un symbole double taille.

Les symboles pour les touches chiffres, le point décimal, le signe égal et les quatre opérations sont évidents. Toutefois nous utiliserons le x minuscule pour la multiplication, car ce symbole est plus utilisé sur les calculettes que le signe * caractérisant la multiplication en BASIC.

Pour la touche de mise en mémoire, nous choisirons un M et pour le rappel de la valeur en mémoire, un R.

Enfin nous créerons deux caractères utilisateur l'un pour le changement de signe (un caractère rassemblant les trois symboles +/-) et l'autre pour la touche d'effacement (un caractère rassemblant les deux lettres EF). Nous aurons ainsi des symboles de même taille que les autres.

Dans un souçi d'esthétique, mais également de clarté de présentation, nous différencierons les touches par des couleurs caractérisant les fonctions, en prenant garde toutefois à n'utiliser, ni le Rouge, ni le Noir, pour les parties de la calculette qui doivent être désignées par le crayon optique.

Les touches chiffres seront en Blanc, le signe = et le point décimal en Jaune, les quatre opérations en Bleu, les deux touches mémoire en Vert, la touche effacement en Cyan (comme le fond d'écran, mais nous placerons cette touche au milieu du tableau), enfin la touche changement de signe en Magenta.

Nous placerons nos touches avec la disposition suivante :

```
7   8   9   M   +
4   5   6   R   -
1   2   3   EF  x
.   0   =  +/-  /
```

Il est bien évident que de nombreuses autres dispositions peuvent être prises. Nous n'avons pas "déposé" le modèle!

Il ne faut toutefois pas croire que nous sommes parvenus à ce choix du premier coup après quelques réflexions sur le papier. De nombreux essais ont été faits pour chercher une disposition des symboles et des couleurs à la fois pratique et plaisante à l'oeil. Un excellent exercice consisterait à chercher une autre disposition peut être plus heureuse, mais rappelons que ce n'est qu'une fois le programme entièrement terminé et testé que l'on peut juger la validité du choix retenu.

Ces principes étant retenus, le programme de tracé est le suivant :

```
10 CLEAR ,,2:DIM OPER$(7),M$(7): DEFDBL A-C,M

20 EF$="                              ":
   M$="L4804RE03SO":SOS$=M$+"L12P"+M$

30 DATA 0,1,2,3,4,5,6,7,-1,-2,-3,-4,-5,-6,-7,-8,
        =,DO,+,RE,-,MI,x,FA,/,SO,M,LA,R,SI

40 READ NOIR%,ROUGE%,VERT%,JAUNE%,
        BLEU%,MAGENTA%,CYAN%,BLANC%,
        NOIRF%,ROUGEF%,VERTF%,JAUNEF%,
        BLEUF%,MAGENTAF%,CYANF%,BLANCF%

50 FOR I=1 TO 7:READ OPER$(I),M$:
   M$(I)="L4803"+M$:NEXT

60 DEFGR$(0)=0,119,68,68,102,68,68,116:
   DEFGR$(1)=1,66,228,72,16,32,71,128
```

```
100 CLS:SCREEN NOIR%,CYAN%,CYAN%

110 BOXF(40,0)-(279,127),BLANCF%

120 BOXF(40,96)-(87,127),JAUNEF%

130 BOXF(136,96)-(183,127),JAUNEF%

140 BOXF(184,0)-(231,63),VERTF%

150 BOXF(184,64)-(231,95),CYANF%

160 BOXF(184,96)-(231,127),MAGENTAF%

170 BOXF(232,0)-(279,127),BLEUF%


200 FOR I=0 TO 5:X=48*I+40:
    LINE(X,0)-(X,127):NEXT

210 FOR I=0 TO 4:Y=32*I:
    LINE(40,Y)-(280,Y):NEXT


300 ATTRB 1,1:COLOR ,BLANC%

305 LOCATE 7,2,0:PRINT "7"

310 LOCATE 13,2:PRINT "8"

315 LOCATE 19,2:PRINT "9"

320 LOCATE 7,6:PRINT "4"

325 LOCATE 13,6:PRINT "5"

330 LOCATE 19,6:PRINT "6"

335 LOCATE 7,10:PRINT "1"

340 LOCATE 13,10:PRINT "2"

345 LOCATE 19,10:PRINT "3"

350 LOCATE 13,14:PRINT "0"
```

```
400 COLOR ,JAUNE%

410 LOCATE 7,14:PRINT "."

420 LOCATE 19,14:PRINT "="

500 COLOR ,VERT%

510 LOCATE 25,2:PRINT "M"

520 LOCATE 25,6:PRINT "R"

600 COLOR ,CYAN%

610 LOCATE 25,10:PRINT GR$(0)

700 COLOR ,MAGENTA%

710 LOCATE 25,14:PRINT GR$(1)

800 COLOR BLANC%,BLEU%

810 LOCATE 31,2:PRINT "+"

820 LOCATE 31,6:PRINT "-"

830 LOCATE 31,10:PRINT "x"

840 LOCATE 31,14:PRINT "/"

850 BOXF(281,0)-(319,31),JAUNEF%:
    ATTRB 0,1:LOCATE 36,2:COLOR ROUGE%,JAUNE%:
    PRINT "M/A":BOX(280,0)-(319,31),NOIR%

900 INPUTPEN X,Y:IF X(0 GOTO 900

910 FOR J=0 TO 10:IF PTRIG GOTO 910 ELSE NEXT J

920 IF X)279 AND Y)0 AND Y(32 GOTO 930
    ELSE GOTO 900

930 PLAY "L9203DO":BOXF(64,144)-(256,176),ROUGEF%
```

*** 10

a/ Réserve deux places mémoire pour deux caractères utilisateur -(M.Ref 32)-.

b/ Réserve 8 places mémoire pour les tableaux de variables chaînes OPER$ et M$ -(M.Ref 46)-.

c/ Les variables dont le nom commence par A,B,C et M seront des variables double précision sauf si un symbole %, ! ou $, ajouté derrière leur nom, change leur type (entier, simple précision ou chaîne de caractères) -(M.Ref 40)-.

*** 20

a/ La chaîne de caractères EF$, composée de 22 espaces est destinée à effacer les chiffres dans la fenêtre d'affichage des résultats.

Cet effacement est nécessaire pour les cas où la chaîne de caractères résultat est plus courte que la chaîne précédemment affichée.

b/ La chaîne M$ est composée de deux notes RE en octave 4 et SO en octave 3. La durée des notes est 48 (ce qui correspond à une blanche).

c/ La chaîne de caractères SOS$ résulte de la concaténation de deux chaînes M$, séparées par la chaîne "L12P" représentant un pause de durée 12 (soit un soupir qui a la durée d'une croche). SOS$ commandée par l'instruction PLAY -(M.Ref 201)- produira au haut parleur un signal "PIN-PON PIN-PON" caractéristique de l'alarme des pompiers. Il sera utilisé pour signaler une erreur.

*** 30 - 50

Il est commode au moment de la mise au point d'une figure en couleurs de disposer de variables portant les noms en clair des couleurs et dont les valeurs sont celles du code de couleurs.

Ainsi la variable NOIR% (variable entière) a la valeur 1 (pour Noir type caractère), la variable BLANCF%, la valeur -8 (pour Blanc type Fond).

La lecture des instructions définissant les couleurs et leur modifications en sont simplifiées. Bien entendu un opérateur confirmé finit par connaître par coeur le code de couleurs et peut se passer d'une telle précaution.

Les sept chaînes OPER$(I) et les sept notes de DO à SI sont associées aux sept touches de fonction, =, +, -, x, /, M et R. Cela nous sera utile pour afficher le marqueur correspondant à la dernière fonction commandée et pour émettre un signal sonore d'identification de la fonction utilisée.

*** 60

Définition des deux caractères utilisateur. GR$(0) est le symbole EF et GR$(1) le symbole +/-.

*** 100

a/ Efface l'écran

b/ Fixe la couleur des caractères au Noir, les couleurs du fond et de l'entourage au Cyan.

*** 110 - 170

Ces instructions définissent les différentes plages rectangulaires avec les couleurs que nous avons choisies pour différencier les touches.

On notera que ces plages sont toutes en couleur fond (voir Annexe I), ceci afin de pouvoir écrire dessus en couleur caractère.

Enfin les touches à gauche et au milieu de la dernière rangée (pour le point décimal et le signe =) sont d'abord tracées en Blanc par l'instruction 110 puis en Jaune par les instructions 120 et 130. Il faut bien entendu prendre garde à ne pas intervertir l'ordre de ces instructions.

*** 200

Tracé de six verticales, entre les ordonnées 0 et 127, depuis l'abscisse 40, avec un espacement horizontal de 48 points.

La couleur, non précisée dans l'instruction LINE -(M.Ref 187)- est, par défaut, la couleur caractère Noir définie en 100 par l'instruction SCREEN

*** 210

Tracé de cinq horizontales, entre les abscisses 40 et 280, depuis l'ordonnée 0, avec un espacement vertical de 32 points.

La couleur est encore la couleur caractère Noir.

*** 300
a/ Impose des caractères de taille double -(M.Ref 26)-.

b/ Fixe la couleur fond des caractères au Blanc.

Il est indispensable de changer la couleur de fond, de Cyan en Blanc, pour pouvoir afficher les symboles chiffres sur les touches blanches. On notera que le premier paramètre de l'instruction COLOR -(M.Ref 189)- n'est pas précisé. C'est donc la couleur Noir, imposée par l'instruction SCREEN en 100, qui est retenue par défaut.

*** 305 - 350
Affiche les dix symboles 0 à 9, en taille double, en Noir sur fond Blanc, sur les dix touches chiffre.

*** 400 - 840
Affichage des différents symboles avec les couleurs fond appropriées.

*** 850
Tracé de la touche "Marche - Arrêt" sur la droite de la calculette.

On notera que la plage Jaune fond définie par l'instruction BOXF a pour abscisse à gauche 281 et non 280, ce qui évite de voir fugitivement disparaître la verticale Noir de droite de la calculette (abscisse 280) entre les ordonnées 0 et 31 avant que l'instruction BOX ne la rétablisse. Un essai avec BOXF(280,0)-(319,31),JAUNEF% montrera l'effet produit.

La touche a une largeur de 5 caractères (soit 40 points). Il est donc possible d'y centrer les trois caractères M/A en taille double en hauteur (instruction ATTRB 0,1 -(M.Ref 26)-).

*** 900
a/ Relève les coordonnées X et Y du point visé par le crayon optique, dès que l'interrupteur du crayon a été fermé -(M.Ref 63)-.

b/ Si les valeurs de X et de Y sont égales à -1, donc négatives, le crayon a été appuyé en dehors de la zone de travail écran. La deuxième instruction de 900 branche alors à la première instruction pour une nouvelle lecture.

*** 910

Nous avons déjà rencontré ce type d'instruction au chapitre V -(M.Bas 75)-.

La variable BASIC PTRIG prend la valeur "Vrai" (-1) si le contact du crayon optique est fermé et la valeur "Faux" (0) dans le cas contraire.

Si le crayon optique est appuyé sur l'écran pendant le déroulement de la boucle en J, celle-ci repart à zéro (branchement en 910 si PTRIG est "Vrai"). Cela impose un certain délai avant que la lecture par le crayon optique ne soit à nouveau effactive et évite des lectures successives dans le cas où l'opérateur fait "rebondir" le crayon sur l'écran.

Or nous allons voir que la désignation de la touche "Marche - Arrêt" met en route la calculette mais que la désignation ultérieure de la même touche conduit à la clôture du programme. Il serait fâcheux qu'un rebond maladroit du crayon conduise à la mise en service et à l'arrêt immédiat du programme !

Les instructions sous l'étiquette 910 n'ont pour rôle que d'éviter cet incident.

*** 920

Si le point désigné par le crayon optique est à l'extérieur de la touche "Marche - Arrêt" le programme est branché en 900 pour un nouveau pointé sinon (ELSE) le programme est branché en 930.

*** 930

a/ Un DO à l'octave 3, de durée 92 (ronde) est émis par le haut-parleur, signalant la mise en service de la calculette.

b/ La fenêtre d'affichage des résultats est tracée en Rouge Fond au dessous de la calculette.

On utilise le Rouge Fond pour pouvoir y inscrire le résultat en couleur caractère.

* * * * * *

Le dessin de la calculette est ainsi terminé. Il est temps d'essayer cette partie du programme pour détecter les erreurs de syntaxe puis juger l'esthétique de la présentation.

Rien n'empêche, à ce stade de la mise au point, de modifier les couleurs ou la taille et la position des touches. De telles modifications, une fois la totalité du programme écrit, seraient beaucoup plus difficiles à faire car elles ont des conséquences sur les instructions suivantes.

Voyons maintenant le programme de calcul proprement dit.

```
1000 COLOR NOIR%,ROUGE%:A=0:B=0:OPER%=1:OPER$=" "

1010 Z$="":PT%=0

1020 GOSUB 5000

1030 ON 1-(U=1 AND V=4)-2*(U=5)-3*(U=3 AND V=4)
     -4*(U=4) GOTO 1100,1200,1300,1400,3000


1100 X=-U*(V=3)-(U+3)*(V=2)-(U+6)*(V=1):
     X$=CHR$(48+X)

1110 IF LEN(Z$)=16+PT% THEN LOCATE 2,20:ATTRB 1,1:
     PRINT "E":PLAY SOS$:LOCATE 2,20:PRINT OPER$:
     ATTRB 0,1:GOTO 1020 ELSE Z$=Z$=X$

1120 IF LEN(Z$)=1 THEN LOCATE 9,20:PRINT EF$

1130 LOCATE 31-LEN(Z$),20:PRINT Z$

1140 GOTO 1020


1200 IF PT%=0 THEN X$=".":PT%=1:GOTO 1110
     ELSE ATTRB 1,1:LOCATE 2,20:PRINT "E":
     PLAY SOS$:LOCATE 2,20:PRINT OPER$:
     ATTRB 0,1:GOTO 1020


1300 IF OPER%=1 THEN OPER%=V+1:A=VAL(Z$):
     ATTRB 1,1:LOCATE 2,20:PRINT OPER$(V+1):
     OPER$=OPER$(V+1):PLAY M$(V+1):ATTRB 0,1

1310 GOTO 1010
```

```
1400 B=A:A=VAL(Z$):ATTRB 1,1:LOCATE 2,20:
     PRINT "=":OPER$="=":ATTRB 0,1:
     ON OPER% GOTO 1410,2000,2100,2200,2300

1410 PLAY SOS$:GOTO 1020


2000 A=A+B: GOTO 2500

2100 A=B-A:GOTO 2500

2200 A=B*A:GOTO 2500

2300 IF A=0 THEN ATTRB 1,1:LOCATE 2,20:
     PRINT "E":PLAY SOS$:LOCATE 2,20:PRINT "/":
     ATTRB 0,1:A=B:GOTO 2500 ELSE A=B/A



2500 Z$=STR$(A):PLAY M$(1)

2510 LOCATE 9,20:PRINT EF$:
     LOCATE 31-LEN(Z$):PRINT Z$

2520 OPER%=1:GOSUB 5000:IF U(4 THEN PLAY SOS$:
     GOTO 2520 ELSE GOTO 1030



3000 ON V GOTO 3100,3200,3300,3400

3100 M=VAL(Z$):A=M:ATTRB 1,1:LOCATE 2,20:
     PRINT "M":PLAY M$(6):ATTRB 0,1:GOTO 2520

3200 Z$=STR$(M):LOCATE 9,20:PRINT EF$:
     LOCATE 31-LEN(Z$),20:PRINT Z$:ATTRB 1,1:
     LOCATE 2,20:PRINT "R":PLAY M$(7):
     ATTRB 0,1:GOTO 1020

3300 A=0:LOCATE 9,20:PRINT EF$:ATTRB 1,1:
     LOCATE 2,20:PRINT GR$(0):ATTRB 0,1:GOTO 1010

3400 ATTRB 1,1:LOCATE 2,20:PRINT GR$(1):ATTRB 0,1:
     C=-VAL(Z$):Z$=STR$(C):LOCATE 31-LEN(Z$),20:
     PRINT Z$:GOTO 1020
```

```
5000 INPUTPEN X,Y:IF X<0 GOTO 5000

5010 FOR I=1 TO 10:IF PTRIG GOTO 5010
     ELSE NEXT:BEEP

5020 IF X>279 AND Y<32 THEN PLAY "L9605SI":
     ATTRB 0,0:COLOR 4,6:CLS:END

5030 IF X<40 OR X>279 OR Y>127 THEN PLAY SOS$:
     GOTO 5000

5040 U=1-(X>87)-(X>135)-(X>183)-(X>231):
     V=1-(Y>31)-(Y>63)-(Y>95)

5050 RETURN
```

*** 1000

a/ Les caractères qui seront maintenant affichés le seront sur fond Rouge (affichage du résulat des opérations dans la fenêtre et affichage du marqueur de type d'opération à gauche de la fenêtre). La couleur caractère est donc fixée au Noir sur fond Rouge

b/ et c/ Les variables A et B (double précision imposée par l'instruction DEFDBL sous l'étiquette 10) sont destinées à contenir les deux opérandes.

d/ La variable OPER% caractérise l'opération effectuée. Elle est initialisée à 1 (touche =). Nous en verrons l'usage plus loin.

e/ La variable OPER$ contient le symbole de la dernière opération commandée.

Elle est utile lorsqu'un message d'erreur "E" ayant été affiché à la place du marqueur d'opération on réaffiche l'opération en cours, après avoir effacé le message d'erreur. Elle est initialisée sous forme d'un espace, puisqu'aucune opération n'a encore été commandée.

*** 1010

a/ La variable Z$ est destinée à recueillir la chaîne de chiffres "frappés" à l'écran à l'aide du crayon optique. Elle est initialisée à "zéro" (chaîne de longueur nulle).

b/ PT% est un "indicateur" qui est mis égal à 1 lorsque la touche point décimal a été désignée. Nous en verrons l'utilité plus loin. Il est initialisé à zéro puisque la touche point décimal n'a pas encore été désignée.

*** 1020
Branchement au sous-programme 5000 de lecture de la touche désignée par le crayon optique.

Nous allons immédiatement étudier ce sous-programme, car son contenu est indispensable à la compréhension de la suite du programme.

*** 5000 - 5010

Relève les coordonnées X et Y du point visé par le crayon optique, dès que l'interrupteur du crayon a été fermé -(M.Ref 63)- (voir les instructions 900 - 910 identiques).

Lorsque les coordonnées X et Y sont acceptées un BIP se fait entendre (instruction BEEP -(M.Ref 30)-)

*** 5020
Si la touche "Marche - Arrêt" a été désignée (X est alors supérieur à 279 ET Y est inférieur à 32) une note SI à l'octave 5 de durée 92 (ronde) est émise, la taille des caractères est fixée à la taille normale, la couleur caractère est fixée au Bleu sur fond Cyan, l'écran est effacé et le programme stoppé par END (Clôture).

*** 5030
Si le point désigné est en dehors de la calculette, le signal d'alarme SOS$ est émis et le programme est branché sur l'étiquette 5000 pour un nouveau pointé.

*** 5040
Le pointé étant dans une zone significative, il reste maintenant à identifier la touche.

Le nombre de cases possibles étant égal à 20 il est impossible d'utiliser l'instruction ONPEN GOTO -(M.Ref 82)- qui n'accepte que huit cases maximum.

a/ U prend les valeurs 1, 2, 3, 4 ou 5 en fonction de X. Il indique le numéro de la case dans la rangée, 1 pour la case de gauche ,5 pour la case de droite.

Supposons en effet que X vaille 123. C'est la case caractère en colonne 15 qui a été désignée, donc la deuxième "case" calculette (rappelons que les valeurs de X renvoyées par le crayon optique sont des multiples de huit augmentés de trois -(M.Bas 77)-. La variable logique (X supérieur à 87) est égale à -1 (Vrai) et les trois autres sont nulles. Ainsi U est égal à 2 ce qui est bien le numéro de la case calculette.

On aurait pu tout aussi bien calculer U par la formule :

U=1+INT((X-40)/48)

puisque les cases calculette ont 48 points de large et que l'abscisse du bord gauche est 40.

Nous avons toutefois indiqué la formule utilisant les variables logiques car elle est générale et fonctionne même si les "touches" ont des tailles inégales, ce qui n'est pas le cas pour la seconde formule.

b/ V d'une façon analogue prend les valeurs 1, 2, 3 ou 4 en fonction de Y. Il indique le numéro de la rangée.

Là encore on aurait pu écrire :

V=1+INT(Y/35)

puisque les "touches" ont 36 points de haut,le haut de la touche 1 ayant l'ordonnée 0.

*** 5050
Instruction de fermeture du sous-programme -(M.Ref 57)-.

Revenons maintenant au programme principal, en retenant que le sous-programme fournit essentiellement les deux valeurs U et V, numéro de la case dans la rangée et numéro de rangée.

*** 1030
Cette instruction teste les valeurs de U et V et branche le programme en conséquence.

On reconnait le mode de branchement par l'instruction ON...GOTO -(M.Ref 80)-, utilisant une variable d'adressage calculée par variables logiques -(M.Bas 83)-.

- Si U = 1 et V = 4 (touche point décimal)
branchement en 1200.

- Si U = 5 (colonne des touches d'opérations)
branchement en 1300

- Si U = 3 et V = 4 (signe égal)
branchement en 1400

- Si U = 4 (colonne mémoire, effacement, changement de signe)
branchement en 3000

- Si aucune des conditions précédentes n'est réalisée (touche chiffre), la variable de branchement vaut 1 et le programme est branché sur 1100.

*** 1100

Le branchement en 1100 est effectué lorsqu'une touche chiffre a été désignée par le crayon optique.

a/ Le chiffre désigné vaut U (c'est à dire le numéro de la case dans la rangée) si la case se trouve dans la rangée 3 où les chiffres marqués sont 1, 2 , 3.

Il vaut U + 3, si la case est dans la seconde rangée où les chiffres marqués sont 4, 5, 6.

Il vaut U + 6 si la case est dans la rangée supérieure où les chiffres marqués sont 7, 8, 9.

Enfin le chiffre désigné est 0 si la rangée est la quatrième (c'est à dire aucune des trois premières).

La variable X calculée à partir des variables logiques (V=3), (V=2), (V=1) -(M.Ref 20)- qui prennent la valeur -1 si la condition entre parenthèses est "Vrai" et la valeur 0 dans le cas contraire, donne donc la valeur du chiffre pointé.

b/ X$ est le caractère représentant le chiffre X.

En effet le numéro de code ASCII de 0 est 48 donc le numéro de code ASCII du nombre X est 48 + X. La fonction CHR$(N) -(M.Ref 113)- donne le caractère de numéro de code ASCII N.

*** 1110
Le nombre de chiffres significatifs d'une variable double précision ne peut dépasser 16.

La variable chaîne de caractères Z$ qui contient la succession des chiffres designés (voir la dernière instruction sous cette étiquette) ne doit donc pas avoir une longueur supérieure à 16, si la chaîne ne contient pas de point décimal et à 17, si la chaîne contient le point décimal. La longeur LEN(Z$) doit donc être au plus égale à 16 + PT% (rappelons que PT% vaut 0 si le point décimal n'a pas été désigné et 1 dans le cas contraire, voir l'étiquette 1200).

En conséquence si la longueur de Z$ est déjà égale à 16 + PT%, on affiche en taille double le caractère E (message d'erreur) en colonne 2, ligne 20 (place du marqueur d'opération) et le signal d'erreur SOS$ est émis.

Le signal d'erreur une fois émis, le symbole OPER$, caractérisant l'opération en cours est à nouveau affiché à la place du marqueur d'opération, et le programme est branché en 1020 pour un nouveau pointé, les caractères ayant été ramenés, par ATTRB 0,1, à la taille double en hauteur utilisée pour l'impression des chiffres dans la fenêtre d'affichage

Si par contre la longueur de Z$ est acceptable (ELSE) la chaîne de caractères Z$ est concaténée à droite avec le caractère X$, pour "ajouter" le chiffre suivant au nombre affiché.

*** 1120
Si la longueur de la chaîne de caractères Z$ est égale à 1, c'est qu'un nouveau nombre est affiché par l'opérateur. Il est donc nécessaire d'effacer le précédent. Dans ces conditions le curseur est placé en ligne 20 (ligne d'affichage des résultats), colonne 9 et la chaîne de caractères EF$ composée de 22 espaces, efface le nombre précédemment affiché.

Il faut noter que la chaîne la plus longue affichable est constituée de 16 digits plus le point décimal plus l'exposant de longueur 4 (la lettre D, le signe de l'exposant et deux digits), soit 21 caractères. La chaîne EF$ efface donc sûrement la chaîne la plus longue.

Nous nous trouvons ici en présence d'une instruction qui n'est en général pas prévue par le programmeur au moment de la conception.

Elle ne s'impose qu'une fois quelques essais de fonctionnement effectués, lorsque l'on constate que cela ne marche pas très bien. C'est alors que la nécessité d'une telle instruction vient à l'évidence! Et encore faut-il faire des essais avec des nombres de plus d'un chiffre, ce que l'on ne fait pas en général en début d'essai. Il est beaucoup plus facile de vérifier que cela "marche" pour une multiplication de 3 par 2 que de 256 par 497!

Cette remarque est destinée aux programmeurs débutants, qui peuvent se sentir "coupables" de n'avoir point prévu cette instruction. Qu'ils soient convaincus que l'auteur du programme n'a rajouté cette instruction qu'après essai malheureux!

De même la chaîne d'effacement EF$ comporte 22 espaces. Or 21 suffiraient d'évidence. Il n'empêche que nous en avons mis 22 "pour le cas où". Une fois le programme mis au point, on hésite bien souvent à "resserrer les tolérances" pour ne pas avoir à revérifier que cela marche dans tous les cas, ce qui imposerait de recommencer de nombreux essais!

*** 1130

a/ La fenêtre d'affichage débute en colonne 8 et se temine en colonne 32 (voir instruction d'affichage de la plage Rouge fond sous l'étiquette 930). On doit y loger une chaîne de 21 caractères au plus (mantisse 16, point décimal 1, exposant 4)

Pour centrer cette chaîne à sa longueur maximale nous placerons le caractère le plus à droite en colonne 30, d'où le numéro de colonne 31 - LEN(Z$) figurant dans l'instruction LOCATE, qui donne bien un numéro de colonne égal à 30 pour un caractère unique.

b/ Affichage de Z$

*** 1140

Branchement en 1020 pour pointé d'une nouvelle case.

Les instructions 1100 à 1140 permettent ainsi de recueillir une chaîne de caractères, représentant le nombre, obtenue par pointés successifs à l'écran. La lecture des chiffres se termine lorsque le pointé est effectué sur une touche qui n'est pas une touche chiffre (étiquette 1030).

*** 1200
Cette instruction est branchée si la touche "point décimal" a été désignée par le crayon optique (U = 1 et V = 4).

Si l'indicateur PT% est égal à 0, la touche "point décimal" n'a pas encore été pointée, du moins pour le dernier nombre en cours d'affichage (dès qu'un nouveau nombre est en cours d'affichage, Z$ est initialisé à zéro, chaîne de longueur nulle et PT% est initialisé à 0, à l'étiquette 1010)

Dans ces conditions (PT% = 0, la touche "point décimal" n'a pas encore été pointée), le branchement en 1200, dû au pointé de la case "point décimal" provoque le basculement à 1 du drapeau PT%, le chargement dans X$ du caractère ".", puis le branchement en 1110, pour concaténation de X$ au bout de la chaîne Z$, affichage de Z$, etc (voir étiquette 1110 et suivantes).

SI PT% n'est pas égal à 0 (le point décimal a déjà été désigné) (instruction ELSE) le message d'erreur "E" est affiché et le signal d'erreur SOS$ émis (voir 1110). Le programme est branché en 1020 pour un nouveau pointé.

*** 1300 - 1310
Cette instruction est branchée si l'une des touches de la colonne opérations (U = 5) a été pointée.

Si le paramètre OPER%, caractérisant la dernière opération pointée est égal à 1 (signe = d'affichage du résultat), alors OPER% est posé égal à V + 1 (soit 2 pour l'addition, 3 pour la soustraction, 4 pour la multiplication et 5 pour la division). Cela permet de mettre en mémoire l'opération demandée.

La variable A est posée égale à la dernière valeur affichée (la fonction VAL(Z$) transforme la chaîne de caractères Z$ en une valeur numérique double précision, puique A a été déclarée comme telle en 10 -(M.Ref 151)-).

Le type d'opération choisie OPER$(V+1), est affiché dans le marqueur d'opérations.

La variable OPER$ est posée égale à OPER$(V+1) pour garder en mémoire le type d'opération choisi en cas d'erreur ultérieure (voir par exemple étiquette 1110 ou 1200).

La note caractérisant le type d'opération, M$(V+1) est émise et le programme branché en 1010 pour le pointé de l'opérande suivant (étiquette 1310)

Si la dernière fonction pointée n'est pas "=", le programme est immédiatement branché en 1010 pour un nouveau pointé. Nous avons décidé en effet de n'accepter une nouvelle opération qu'après affichage du résultat de l'opération précédente par la commande "=".

Cette dernière option n'est pas classique, mais rien ne nous interdit de la choisir.

*** 1400 - 1410
Cette instruction est branchée si la touche "=" a été pointée (U = 3 et V = 4)

a/ La valeur numérique contenue dans A est transferrée dans B.

b/ La dernière valeur affichée (chaîne Z$) est placée dans A.

c/ d/ et e/ Le marqueur d'opération affiche le signe "=".

f/ La variable chaîne de caractère OPER$ est posée égale à "=".

g/ En fonction du paramètre OPER%, caractéristique du type d'opération, le programme est branché en 2000 (OPER%=2, addition), 2100 (OPER%=3, soustraction), 2200 (OPER%=4, multiplication) ou 2300 (OPER%=5, division).

Si la dernière opération commandée était un affichage de résultat (OPER%=1), le programme est branché en 1410 pour émission d'un signal d'erreur et retour à un nouveau pointé.

*** 2000 - 2300
L'opération commandée est exécutée sur les deux opérandes A et B, le résultat est stocké dans A, puis le programme branché en 2500.

Dans le cas particulier de la division (2300), la valeur nulle du diviseur A est testée pour éviter que le programme ne se "plante". Si A est nul, le message d'erreur est affiché et le signal d'erreur émis, puis A est de nouveau posé égal à B pour recommencer la division.

*** 2500

a/ Le résultat A de l'opération est transformé en chaîne de caractères par la fonction STR$(A) -(M.Ref 147)-

b/ La note caractéristique de l'opération "=" est émise.

*** 2510

La dernière valeur affichée est effacée et le résultat (chaîne Z$) est affiché.

Le résultat figurant sous forme de Z$ peut alors être utilisé pour une opération ultérieure.

*** 2520

a/ La variable OPER% est posée égale à 1 (caractéristique de l'opération = d'affichage du résultat) ce qui permet de commander une nouvelle opération en 1300, en utilisant le résultat comme premier opérande.

b/ Branchement au sous-programme 5000 de relevé de la case pointée.

c/ Si le pointé n'a pas été fait dans l'une des deux dernières colonnes des touches de la calculette, un signal d'erreur est émis et le programme rebranché en 2520 pour un nouveau pointé. Cela interdit de rentrer un nouvel opérande en pointant les touches chiffres.

Sinon (ELSE) le programme est branché en 1030 pour analyse de la touche pointée.

*** 3000

Le programme est branché sur cette étiquette si U = 4, c'est à dire si une touche de la quatrième colonne a été pointée.

L'instruction ON V GOTO -(M.Ref 80)- branche le programme sur les instructions 3100 (V = 1, touche de mise en mémoire), 3200 (V = 2, touche de rappel de la valeur en mémoire), 3300 (V = 3, touche d'effacement) ou 3400 (V = 4, touche de changement de signe).

*** 3100

La touche M de mise en mémoire a été pointée.

La valeur du nombre affiché (VAL(Z$), -(M.Ref 151)-) est stockée dans la variable M (double précision imposée en 10).

Cette valeur est également stockée dans la variable A (le pointé de la touche "mise en mémoire" est une commande qui signifie, comme les autres commandes d'opération que le pointé des chiffres est terminé et que le nombre affiché est "complet").

Le marqueur d'opération affiche M et la note correspondant à l'opération, M$(5), est émise. Le programme est rebranché sur 2520 comme après une opération d'affichage de résultat.

*** 3200

La touche R de rappel de la valeur en mémoire a été pointée.

La variable chaîne de caractères Z$ est chargée avec le contenu de la mémoire M -(M.Ref 151)- et la chaîne Z$ affichée, après effacement de la dernière valeur.

Le merqueur d'opération affiche R et la note correspondant à l'opération, M$(7) est émise.

Le programme est branché en 1020 pour un nouveau pointé.

On remarquera que la valeur rappelée peut être utilisée comme premier ou comme deuxième opérande.

Il faut donc brancher le programme en 1020 pour un nouveau pointé et non en 2520 comme après une mise en mémoire, car dans ce cas l'affichage du résultat par la commande "=" est impossible.

*** 3300

La touche EF d'effacement a été pointée.

La variable A est mise à zéro, le symbole GR$(0) (EF) est affiché sur le marqueur d'opérations et le programme branché en 1010 pour lecture d'un nouvel opérande.

*** 3400
La touche +/- de changement de signe a été pointée.

Le symbole GR$(1) (+/-) est affiché sur le marqueur d'opérations.

La valeur de la chaîne VAL(Z$) -(M.Ref 151)-, changée de signe est placée dans la variable double précision C.

La variable C est transformée en chaîne de caractère Z$ par la fonction STR$(C) -(M.Ref 147)- et la nouvelle chaîne Z$ affichée.

Le programme est branché en 1020 pour un nouveau pointé.

(On aurait pu simplifier les instructions de changement de signe en écrivant directement Z$=STR$(-VAL(Z$)), sans passer par l'intermédiaire de la variable C.)

La mise au point d'un tel programme est délicate, car il est nécessaire de faire de nombreux essais pour tenter de passer en revue toutes les combinaisons de commandes qui peuvent conduire à un comportement aberrant de la calculette.

Il est donc recommandé de ne progresser que pas à pas dans la programmation, en essayant une commande ou une fonction (lecture d'un nombre, opérations, mémoire etc) à fond avant de programmer une autre fonction. De même on aura intérêt à placer des instructions provisoires aux étiquettes branchées par les ON...GOTO, pour vérifier que les branchements se font bien comme prévus sur pointé des touches écran.

Par exemple il est indispensable de vérifier le sous-programme 5000 en affichant provisoirement les valeurs de U et V correspondant au pointé des différentes touches. Il est évident que si les valeurs de U et V ne sont pas les valeurs attendues, par suite d'une erreur de logique ou d'une erreur matérielle (valeur numérique fausse en 5040 par exemple), le pointé d'une touche de fonction peut avoir un résultat curieux, même si le programme correspondant à cette commande est correct.

Un excellent exercice consisterait à compléter la calculette en programmant des fonctions complémentaires, racine carrée, fonctions trigonométriques, fonctions exponentielle et logarithme, affichage avec un nombre fixe de décimales, etc.

9

Tableau de données

Nous supposerons que nous disposons des chiffres d'affaire de trois départements d'une société, réalisés chaque jour de la semaine. Nous voulons les mettre en mémoire, les présenter sur un tableau rectangulaire de trois colonnes A, B et C (pour les départements) et de cinq colonnes pour les jours de la semaine et calculer les totaux, par département et par jour, ainsi que les pourcentages correspondants par rapport au total général.

Nous allons bâtir un tableau de 8 lignes et de 6 colonnes. La première colonne servira à l'identification des lignes et la première ligne à l'identification des colonnes. Il se présentera donc de la façon suivante :

	0	1	2	3	4	5
0		A	B	C	TOTAL	%
1	LUN					
2	MAR					
3	MER					
4	JEU					
5	VEN					
6	TOT					
7	%					

Nous avons numéroté le lignes de 0 à 7 et les colonnes de 0 à 5, les ligne et colonne 0 étant destinées aux renseignements.

Les trois colonnes 1 à 3 et les cinq lignes 1 à 5 du tableau seront remplies par l'opérateur.

Les totaux correspondant à chacun des départements seront affichés en ligne 6, colonnes 1 à 3, les pourcentages en ligne 7, colonne 1 à 3.

Les totaux correspondant à chacun des jours de la semaine seront affichés en colonne 4, lignes 1 à 5, les pourcentages en colonne 5, lignes 1 à 5.

Le total général sera affiché en ligne 6, colonne 4

La case en haut à gauche et les trois cases en bas à droite sont disponibles.

La colonne 0 du tableau aura une largeur de 40 points d'écran de façon à pouvoir y centrer 3 caractères.

La colonne 5 destinée à recevoir des pourcentages aura 48 points de large de façon à pouvoir y centrer 4 caractères. Les pourcentages seront en effet donnés avec une décimale, ce qui donne 4 caractères, compte tenu du point décimal.
Dans le cas particulier où nous aurons un pourcentage égal à 100, nous inscrirons la chaîne "Cent" au lieu de la valeur numérique, ce qui donne toujours 4 caractères.

Il nous reste donc 40 - 5 - 6 = 29 cases caractères de huit points pour les colonnes 1 à 4. Chacune des colonnes 1 à 3 aura donc une largeur de 7 x 8 = 56 points et la colonne 4, pour les totaux, aura une largeur de 8 x 8 = 64 points.

Enfin les 8 lignes du tableau auront un hauteur de trois caractères, soit 3 x 8 = 24 points, ce qui occupe 24 lignes caractère de l'écran.

Pour remplir avec les données, le tableau trois colonnes sur cinq lignes, nous pourrions décider d'opérer, a priori, ligne par ligne ou au contraire colonne par colonne. Ce choix dépend de la façon dont l'opérateur dispose de ses données, par département ou par jour. Il nous est donc difficile de faire un choix qui ne soit pas arbitraire et qui suppose un type particulier de recueil de données qui peut varier d'une organisation à l'autre.

Nous laisserons donc la possibilité à l'opérateur de choisir la solution qu'il préfère.

Pour cela nous le laisserons maître de choisir la case à remplir en déplaçant un "spot" sur la grille 3 x 5 à l'aide des quatre flèches de commande du clavier.

L'affichage d'un nombre dans la case désignée par le "spot", se fera en tapant successivement, au clavier, les chiffres composant ce nombre. Ces chiffres apparaitront dans la case sur la marge de droite, les chiffres précédemment tapés se déplaçant vers la gauche. Le nombre ainsi tapé se présentera constamment justifié à droite.

Par ailleurs, compte tenu de la taille des cases, limitée par la taille même de l'écran, nous n'accepterons que des nombres entiers positifs ayant au plus cinq chiffres (les cases ont 7 caractères de large). Il faudra donc prendre garde à faire respecter automatiquement cette contrainte et à interdire la frappe de caractères autres que des chiffres.

Compte tenu de ces dispositions, le programme de dessin du tableau se présente de la façon suivante :

```
10 SCREEN 4,6,6

20 CLS:CLEAR

30 DEFINT N:DIM Z(3,5),N(3,5)

40 M$="######":T$="######":S$="########":P$="##.#"
   IP$=" "+T$+" "+T$+" "+T$+" "+S$+" ###.#":
   PP$=P$+"    "+P$+"    "+P$

50 DO$="L4804DO"

100 BOXF(0,24)-(39,191),-6

110 BOXF(40,0)-(319,23),-6

120 BOXF(0,0)-(39,24),-5

130 BOX(0,0)-(319,191)

140 LINE(40,0)-(40,190):FOR X=98 TO 210 STEP 56:
    LINE(X,0)-(X,190):NEXT:LINE(272,0)-(272,190)

150 FOR Y=24 TO 168 STEP 24:LINE(0,Y)-(319,Y):NEXT
```

```
200 COLOR 7,5:LOCATE 8,1,0:PRINT "A":
    LOCATE 15,1:PRINT "B":
    LOCATE 22,1:PRINT "C":
    LOCATE 28,1:PRINT "TOTAL"
    LOCATE 36,1:ATTRB 1,0:PRINT "%":ATTRB 0,0

210 DATA LUN,MAR,MER,JEU,VEN,TOT," %"

220 FOR J=4 TO 22 STEP 3:
    READ Z$:LOCATE 1,J:PRINT Z$:NEXT
```

*** 10

Fixe la couleur caractères au Bleu (4), la couleur du fond et de l'entourage au Cyan (6).

*** 20

a/ Efface l'écran

b/ Met à zéro toutes les variables et récupère la place des variables simples et des tableaux.

Cette instruction n'est pas nécessaire en début de programme, mais elle sera utile pour recommencer un nouveau tableau (remise rapide à zéro de toutes les variables) .Il faut prendre garde toutefois à dimensionner à nouveau les tableaux par DIM (une instruction DIM avant un CLEAR est sans effet puisqu'annulée par le CLEAR).

*** 30

a/ Toutes les variables dont le nom commence par un N seront considérées comme des variables entières -(M.Ref 40)-

b/ Les deux variables Z (simple précision) et N (entier) sont déclarées comme étant des tableaux dépendant de deux paramètres, le premier pouvant varier de D à 3, le second de D à 5.

La variable Z servira de stockage des données, Z(I,J) étant la donnée inscrite dans la case de la colonne I et de la ligne J.

Les valeurs I=D et J=D sont inutilisées ce qui contitue une réservation abusive de neuf places mémoires, mais il est plus commode de raisonner avec des indices commençant à 1 plutôt qu'à 0.

La variable N servira d'indicateur signalant qu'une donnée à été entrée.

Si une donnée a été entrée dans la case en colonne I, ligne J, N(I,J) sera posé égal à -1 (voir étiquette 2100).

*** 40
Les variables définies ici seront utilisées comme images dans les instructions PRINT USING -(M.Ref 90)-. Nous en verrons la justification au moment de l'emploi.

*** 50
La variable OO$ sera utilisée avec une instruction PLAY -(M.Ref 201)- pour émettre un OO de durée 48 (blanche) à l'octave 4.

Nous en arrivons maintenant au dessin proprement dit du tableau. Pour clarifier le listing, nous commencerons cette partie du programme à l'étiquette 100.

*** 100 - 110
Colorent en couleur Magenta fond (-6) la colonne 0 et la ligne 0 du tableau.

*** 120
Colore en Bleu fond (-5) la case inutilisée en haut et à gauche du tableau.

*** 130
Trace un cadre autour du tableau en couleur caractère Bleu imposée par l'instruction SCREEN en 10 (valeur prise par défaut puisque non précisée).

*** 140
a/ Trace la verticale séparant les colonnes 0 et 1 du tableau

b/ Trace les trois verticales de séparation des colonnes 1, 2, 3 et 4 du tableau.

c/ Trace la verticale de séparation des deux dernières colonnes du tableau.

On notera que les abscisses des verticales ne sont pas des multiples exacts de huit, mais des multiples de huit augmentés de 2.

Ceci permettra de ne pas "coller" le caractère écrit à droite de la case sur la verticale de séparation.

*** 150
Trace les sept horizontales de séparation des lignes du tableau.

*** 200
Affiche les symboles d'identification des colonnes sur la ligne 0 du tableau, donc sur la ligne 1 de l'écran en Blanc (7) sur fond Magenta (5).

Pour centrer le symbole % au milieu de la dernière colonne qui a une largeur de six caractères (48 points), on l'affiche en largeur double (ATTR@ 1,0). La taille des caractères est ramenée à la normale par la dernière instruction ATTR@ 0,0.

On notera que la première instruction LOCATE se termine par un 0 en troisième paramètre pour effacer le curseur dont nous n'avons pas besoin.

*** 210 - 220
Affiche les symboles d'identification des lignes dans la colonne 0 du tableau.

On notera que lors des tracés successifs des divers tableaux, le programme passe plusieurs fois sur l'instruction 210 de lecture des DATA stockées en 200. On pourrait craindre une erreur 0D (OUT of DATA), venant du fait que les DATA ont déjà été lues et que le pointeur est en fin de liste. Comme la reprise d'un tableau se fait (instruction 3590) en repassant par l'instruction 20 (CLEAR), le pointeur est remis à zéro sans qu'il ne soit nécessaire de l'imposer par une instruction RESTORE

Les instructions de 10 à 220 constituent un "bloc exécutable" que l'on doit tester avant de passer à la suite du programme. Nous ne saurons trop insister sur cette méthode de vérification, qui constitue l'une des révolutions du micro par rapport aux méthodes classiques de la grosse informatique.

Si le programme "marche" et vous satisfait sur le plan de l'esthétique, profitez en pour le sauver sur une cassette, une panne de secteur est si vite arrivée! Si vous disposez également d'une imprimante, faites donc un listing. Bien entendu, si vous avez recopié fidèlement les instructions de ce livre cette précaution est inutile. Mais si vous avez apporté quelques améliorations, il est très commode de disposer d'un listing à jour avant d'attaquer la suite.

Rappelons que pour lister un programme sur l'imprimante, la commande générale s'écrit :

```
LIST "LPRT:(N)",E1-E2
```

où N est le nombre de caractères par ligne disponibles sur l'imprimante, E1 la première étiquette à lister et E2 la dernière.

Pour l'imprimante thermique PR 90 040, N maximal est égal à 40. Il n'est pas nécessaire dans ce cas de faire figurer (40) derrière les deux points, la valeur 40 étant retenue par défaut. Mais il ne faut pas oublier les guillemets ni les deux points.

Pour l'imprimante à aiguilles PR 90 080, N maximal est égal à 80. Il faut dans ce cas préciser la valeur de N (sauf si l'on veut lister sur 40 colonnes seulement, ce qui n'est pas économique!).

Enfin si l'on désire tout le listing, il suffit de ne pas préciser les étiquettes E1 et E2 (on supprime dans ce cas la virgule après le dernier guillemet).

* * * * * *

Programmons maintenant l'introduction des données.

Rappelons que nous désirons afficher un "spot" dans la case où nous allons entrer la donnée, en déplaçant le "spot" à l'aide des quatre Flèches de commande du clavier.

Le "spot" sera constitué d'un rectangle de 5 cases caractère en Rouge fond de telle sorte que l'on puisse écrire dessus pour entrer la donnée et le supprimer en passant sa couleur au Cyan fond (en réaffichant la donnée si elle existe).

Il nous faut prévoir une commande pour signaler que nous avons entré la totalité des chiffres constituant la donnée (les chiffres une fois tapés s'affichent à droite de la case du tableau en repoussant les précédents vers la gauche). Cette commande est tout naturellemebt la touche ENTREE.

Il faut enfin prévoir une commande signalant la fin de la phase d'entrée des données.

On pourrait envisager de passer au calcul des sommes et des pourcentages de façon automatique dès que le tableau est entièrement rempli (il n'y a plus alors d'indicateurs N(I,J) nuls). Mais il est inutile de remplir le tableau avec des valeurs nulles s'il en existe et il peut être commode, avant de déclancher le calcul de vérifier le tableau et y apporter, si nécessaire, des corrections.

Nous choisirons donc la solution manuelle, avec commande du calcul par la touche ACC par exemple. Il faudra alors prévoir de compléter le tableau avec des zéros, si toutes les cases n'ont pas été remplies.

Pour l'instant programmons l'entrée des données avec le déplacement du "spot".

```
300 IA=1:JA=1:CA=7:LA=4:
    BOXF(8*CA,8*LA)-(8*CA+39,8*LA+7),-2


500 Z$=INPUT$(1):Z=ASC(Z$):W=1-(Z=8)-2*(Z=9)
    -3*(Z=10)-4*(Z=12)-5*(Z)47 AND Z(58)
    -6*(Z=13)-7*(Z=22)

510 ON W GOTO 500,600,700,800,900,2000,2100,3000


600 IN=IA-1:IF IN=0 THEN IN=1

610 JN=JA:GOTO 1000

700 IN=IA+1:IF IN=4 THEN IN=3

710 JN=JA:GOTO 1000
```

```
800 JN=JA+1:IF JN=6 THEN JN=5

810 IN=IA:GOTO 1000

900 JN=JA-1:IF JN=0 THEN JN=1

910 IN=IA


1000 CN=7*IN:LN=3*JN+1:ZZ$=""

1010 BOXF(8*CA,8*LA)-(8*CA+39,8*LA+7),-7

1020 IF N(IA,JA) THEN LOCATE CA,LA:COLOR 4,6:
     PRINT USING M$;Z(IA,JA)

1030 BOXF(8*CN,8*LN)-(8*CN+39,8*LN+7),)2:
     IF NOT N(IN,JN) GOTO 1050

1040 COLOR 4,1:LOCATE CN;LN:
     PRINT USING M$;Z(IN,JN)

1050 IA=IN:JA=JN:CA=CN:LA=LN:GOTO 500


2000 ZZ$=ZZ$+Z$:IF LEN(ZZ$)=5 GOTO 2100

2010 Z=VAL(ZZ$):COLOR 7,1:LOCATE CA,LA:
     PRINT USING M$;Z:GOTO 500

2100 Z(IA,JA)=VAL(ZZ$):N(IA,JA)=-1:COLOR 4,3:
     LOCATE CA,LA:PRINT USING M$;Z(IA,JA):
     ZZ$="":PLAY DO$:GOTO 500
```

*** 300

Nous noterons IA et JA les numéros de colonne et de ligne de la case du tableau où se trouve actuellement le spot et CA et LA les numéros correspondants de colonne et de ligne écran où doit être affichée la donnée dans la case.

On voit facilement que pour la case IA=1 et JA=1, case en haut et à gauche du tableau, la donnée doit être affichée par un PRINT USING, avec une image M$ comportant 5 caractères dièse, en colonne CA=7 et en ligne LA=4. Dans ces conditions la donnée s'affiche sur la ligne au centre de la case avec le chiffre des unités situé sur le bord droit de la case en colonne écran 11.

Comme les cases du tableau ont une largeur de 7 caractères et une hauteur de 3 caractères, les relations entre IA et CA et entre JA et LA sont CA=7*IA et LA=3*JA+1.

Les quatre premières instructions de 300 initialisent donc IA, JA, CA et LA, aux valeurs correspondant à la case en haut et à gauche.

La cinquième instruction trace un rectangle en couleur fond Rouge (-2) occupant la place de cinq caractères, avec le premier caractère en colonne 7 et en ligne 4 écran. Ce rectangle constitue le "spot", placé initalement dans la case haut-gauche du tableau.

Puisque l'on trace un rectangle plein en couleur fond sur fond, on notera que l'on aurait pu mettre comme abscisse du coin bas-droit toute valeur comprise entre 8*CA+32 et 8*CA+39 (voir Annexe I sur la gestion des couleurs et les commentaires de l'étiquette 1D10).

*** 500

a/ Le programme se met en attente de la frappe d'une touche (et d'une seule) au clavier, puis place le caractère correspondant dans Z$ -(M.Ref 174)-

b/ Z est le numéro de code ASCII de Z$ et caractérise la touche frappée.

c/ Calculée à partir de variables logiques de la forme (Z=N), la variable W prend les valeurs suivantes :

2 si la Flèche Gauche a été tapée (numéro de code 8)

3 si la Flèche Droite a été tapée (numéro de code 9)

4 si la Flèche Bas a été tapée (numéro de code 10)

5 si la Flèche Haut a été tapée (numéro de code 11)

6 si la touche tapée est une touche chiffre
(numéro de code entre 48 pour "0" et 57 pour "9")

7 si la touche ENTREE a été tapée (numéro de code 13)

8 si la touche ACC a été tapée (numéro de code 22)

1 pour toute autre touche.

*** 510

Branche aux étiquettes placées à droite du GOTO -(M.Ref 80)- suivant les valeurs de W calculées à l'instruction précédente.

-Si aucune touche utile n'a été frappée (W = 1), le programme est rebranché en 500 pour une nouvelle lecture.

-Si une des flèches a été frappée, branchement à l'une des étiquettes 600 à 900, pour déplacement du "spot".

-Si un chiffre a été tapé, branchement en 2000.

-Si la touche ENTREE a été frappée, branchement en 2100 (commande de fin d'entrée d'une donnée)

-Si la touche ACC a été frappée, branchement en 3000 (commande de début de calcul des sommes et des pourcentages).

*** 600 - 910

Ces instructions calculent les valeurs IN et JN (nouvelle colonne et nouvelle ligne de la case, commandées par l'action sur l'une des quatre flèches du clavier). Les commandes intempestives risquant de faire sortir le "spot" de la grille 3 x 5 cases, sont interdites par les instructions IF.

Par exemple, la frappe de la Flèche Droite (numéro de code 9) ayant provoqué le branchement en 700, la nouvelle valeur de I, soit IN est égale à IA+1, sauf si IN atteint la valeur interdite 4, auquel cas IN est posée égale à 3. Le programme est ensuite branché en 1000, pour le déplacement du "spot".

*** 1000

a/ et b/ Calcul de la position CN, LN (numéro de colonne et de ligne écran) où devrait être affichée la prochaine donnée, si elle est entrée par l'opérateur.

Nous avons vu la relation entre les C et les I d'une part, les L et les J d'autre part à l'occasion des commentaires de l'étiquette 300.

c/ La variable ZZ$ destinée à recueillir les chiffres successifs composant la prochaine donnée est mise à "zéro" (chaîne de caractères de longueur nulle).

*** 1010

La plage Rouge fond tracée à l'ancienne position du "spot", caractérisée par CA et LA, est affichée en Cyan fond (-7). Le Rouge s'efface donc, le "spot" tracé alors en Cyan fond sur fond Cyan disparaissant.

Le fait de tracer le rectangle plein jusqu'à l'abscisse 8*CA+39 en Cyan fond, impose à tous les points de la plage correspondant à cinq caractères dont le premier est en colonne CA, de passer en points de type couleur fond. Cela conduit donc à l'effacement de tous les caractères affichés sur cette plage.

Si nous avions programmé une abscisse du coin droit égale à 8*CA+32, nous aurions bien imposé aux points fond de la dernière case caractère de passer du Rouge fond au Cyan fond mais les points de type caractère, c'est-à-dire les points du dernier caractère, n'auraient pas été affectés par cet ordre et seraient restés affichés, avec leur couleur caractère. On notera que la première "colonne" de huit points à gauche d'une case caractère n'est jamais occupée par des points caractères. Changer le type de couleur de ces points pour une couleur fond différente, provoque le changement de couleur de tous les points fond de la case, mais n'influe pas sur le caractère lui-même.

Toutes ces remarques peuvent sembler subtiles. Il n'empêche que le remplacement de 39 par 32, qui a priori fait le même effet, peut conduire à des comportements du programme difficilement compréhensibles. Pour en juger, essayez de remplacer 39 par 32!

Inutile de préciser que l'auteur n'a découvert toutes ces subtilités qu'après quelques expériences malheureuses.

*** 1020

Si la case IA, JA, c'est-à-dire la case qui vient d'être abandonnée par le "spot" a déjà été remplie par une donnée, l'indicateur N(IA,JA) est égal à -1.

La variable N(IA,JA) considérée comme une variable logique est donc "Vrai" (c'est pourquoi nous la posons égale à -1 si la case est occupée, voir étiquette 2100). La condition derrière l'instruction IF est alors "Vrai" et les instructions derrière l'instruction THEN exécutées sinon le programme passe à l'étiquette suivante.

Ainsi, dans la mesure où une donnée a déjà été entrée dans la case IA, JA, la donnée Z(IA,JA) est affichée en Bleu (4) sur fond Cyan (6) par l'instruction PRINT USING.

*** 1030

a/ Le "spot" est affiché en Rouge fond à la nouvelle place.

b/ Si la nouvelle case occupée par le spot n'a pas déjà été remplie alors l'indicateur N(IN,JN) est égal à 0. Dans ces conditions la variable logique N(IN,JN) est égale à "Faux" et NOT N(IN,JN) est égale à "Vrai". La condition derrière l'instruction IF est "Vrai" et le programme saute en 1050, évitant 1040.

*** 1040

Cette instruction n'est exécutée que si la nouvelle case occupée par le "spot" a déjà été remplie. Dans ces conditions la donnée Z(IN,JN) est affichée en Bleu (4) sur fond Rouge (1). Ainsi le passage du "spot" sur une donnée déjà affichée ne l'efface pas. Si aucune donnée ne figurait dans la case, le "spot" reste Rouge sans inscription.

On notera qu'en l'abscence d'utilisation d'indicateur marquant l'existance de donnée, on serait conduit à afficher un 0 au passage du "spot".

On aurait pu penser faire un test sur la valeur nulle ou non de Z(IN,JN). Mais il aurait été alors impossible de faire la différence entre une donnée nulle et une donnée non entrée!

Là encore il ne faut pas croire que toutes ces subtilités ont été prévues et programmées a priori par l'auteur.

Ce n'est qu'après quelques essais de fonctionnement, que le test par la valeur nulle ou non de la donnée s'est révélé défectueux et que la nécessité d'utiliser un indicateur est apparue évidente.

*** 1050

Les nouvelles valeurs IN, JN, CN, LN deviennent désormais les valeurs actuelles, IA, JA, CA, LA. D'où les quatre instructions de réinitialisation avant le retour en 500 pour lecture d'une nouvelle commande.

*** 2000

L'instruction 2000 est branchée si la lecture en 500 est celle d'un chiffre.

La variable Z$ contient donc un chiffre qui doit venir à la droite de la chaîne de chiffres déjà entrée (stockée dans ZZ$ mis à zéro à chaque déplacement du "spot", voir étiquette 1000, ou après lecture complète d'une donnée, voir étiquette 2100).

a/ Z$ est concaténée à droite de ZZ$

b/ Si la longueur LEN(Z$) -(M.Ref 128)- atteint la valeur 5 (valeur maximale décidée, a priori, du fait du dessin du tableau), le programme est branché en 2100, c'est-à-dire à l'étiquette de branchement imposée par la commande ENTREE de fin d'entrée de donnée. Il est donc impossible d'entrer un nombre de plus de cinq chiffres.

*** 2010

a/ Z est la valeur numérique VAL(ZZ$) -(M.Ref 151)- de la chaîne partielle ZZ$, constituée des caractères de la chaîne représentant les premiers chiffres entrés.

b/ c/ et d/ Affichage de la valeur Z, cadrée à droite par l'instruction PRINT USING, puisque l'image M$ contient 5 dièses, qui placent le chiffre des unités de Z en colonne CA+4. On voit ainsi s'afficher successivement les chiffres avec le dernier frappé, cadré à droite. La couleur de caractères est Blanc (7) sur fond Rouge (1),couleur fond du "spot"

f/ Branchement en 500 pour une lecture d'un nouveau chiffre ou d'une commande.

*** 2100

Les instructions sous cette étiquette sont branchées, soit que la touche ENTREE ait été commandée (voir 510), soit que le nombre de chiffres entrés ait atteint cinq.

a/ La valeur numérique correspondand à la chaîne ZZ$, VAL(ZZ$) -(M.Ref 151)-, est stockée dans la variable Z(IA,JA) (IA et JA sont les valeurs colonne et ligne de la case du tableau actuellement occupée par le "spot").

b/ L'indicateur N(IA,JA) est posé égal à -1, puisque la case (IA,JA) est maintenant occupée.

c/ d/ et e/ Affichage en Bleu (4) sur fond Jaune(3) de la donnée, cadrée avec le chiffre des unités à droite.

Le "spot" semble passer au Jaune puisque les chiffres s'affichent sur un fond Jaune. Ce changement de couleur confirme que la donnée est entrée.

f/ La chaîne ZZ$ est remise à zéro, pour une nouvelle lecture.

g/ Le haut-parleur émet un DO -(M.Ref 201)-, confirmant l'entrée d'une donnée.

h/ Branchement pour lecture d'un nouveau chiffre (pour correction de la donnée) ou d'une commande.

Il est recommandé d'essayer dès maintenant cette partie de programme. Il est évident que ce type de programme contient de nombreux pièges potentiels, nous en avons décrits quelques uns, et qu'il est indispensable de l'essayer avant d'aller plus loin.

* * * * * *

Etudions maintenant le calcul des sommes et des pourcentages, déclanché par la commande ACC, qui branche le programme en 3000 (voir étiquette 510).

```
3000 BOXF(8*CA,8*LA)-(8*CA+39,8*LA+7),-7:
     COLOR 4,6:LOCATE CA,LA:
     PRINT USING M$;Z(IA,JA)

3010 FOR I=1 TO 3:FOR J=1 TO 5:
     IF NOT N(I,J) THEN LOCATE 7*I+4,3*J+1:
     PRINT "0"

3020 NEXT J,I

3030 COLOR 1

3040 FOR I=1 TO 3

3050 S=0:FOR J=1 TO 5:S=S+Z(I,J):NEXT J:SV(I)=S

3060 LOCATE 7*I-1,19:PRINT USING T$;S:NEXT I
```

```
3070 FOR J=1 TO 5

3080 S=0:FOR I=1 TO 3:S=S+Z(I,J):NEXT I:SH(J)=S

3090 LOCATE 28,3*J+1:PRINT USING T$;S:NEXT J


3200 S=0:FOR I=1 TO 3:S=S+SV(I):NEXT

3210 LOCATE 27,19:PRINT USING S$;S


3300 COLOR 0

3310 FOR I=1 TO 3

3320 P=100*SV(I)/S:LOCATE 7*I,22:
     IF P=100 THEN COLOR 1:
     PRINT "Cent":COLOR 0:GOTO 3340

3330 PRINT USING P$;P

3340 PV(I)=P:NEXT


3400 FOR J=1 TO 5

3410 P=100*SH(J)/S:LOCATE 35,3*J+1:
     IF P=100 THEN COLOR 1:
     PRINT "Cent":COLOR 0:GOTO 3430

3420 PRINT USING P$;P

3430 PH(J)=P:NEXT


3500 BOXF(210,168)-(319,191),4:
     BOXF(272,144)-(319,168),4

3510 COLOR 1,7:LOCATE 29,22:PRINT "Tapez RAZ"

3520 R$=INPUT$(1):IF ASC(R$)<>12 GOTO 3520

3530 COLOR 4,6:CLS

3540 LOCATE 5,10:PRINT "Pour"
```

```
3550 LOCATE 5,12:
     PRINT "Imprimer le tableau    tapez ":
     COLOR 7,1:PRINT "I":COLOR 4,6

3560 LOCATE 5,14:
     PRINT "Taper un autre tableau tapez ":
     COLOR 7,1:PRINT "T":COLOR 4,6

3570 LOCATE 5,14:
     PRINT "Cloturer               tapez ":
     COLOR 7,1:PRINT "I":COLOR 4,6

3580 R$=INPUT$(1):
     ON 1-(R$="I")-2*(R$="T")-3*(R$="C")
     GOTO 3580,4000,20,5000

5000 SCREEN 4,6,6:CLS
```

*** 3000

Les instructions sous cette étiquette sont branchées lorsque l'opérateur a tapé la commande ACC (voir 510), pour déclancher le calcul des sommes et des pourcentages.

La dernière donnée entrée a laissé la donnée affichée en Bleu sur le "spot" en Jaune (voir étiquette 2100). Il est donc nécessaire de réafficher la dernière donnée comme les précédentes.

a/ Passe le "spot" en couleur Cyan fond (-7).

b/ Fixe la couleur caractère au Bleu (4) sur fond Cyan (6).

c/ et d/ Affiche la donnée de la dernière case IA,JA en CA,LA.

*** 3010 - 3020

Il se peut que certaines cases n'aient pas reçu de données. On sait que ces cases sont caractérisées par un indicateur N(I,J) nul.

Avant d'effectuer les sommations, il est nécessaire de compléter le tableau par des zéros dans les cases non remplies (on peut être d'un avis contraire et décider de ne pas compléter ainsi le tableau, auquel cas il suffit de sauter les étiquettes 3010 et 3020).

Ouverture de deux boucles emboitées en I et J. Si N(I,J) est égal à zéro (pas de donnée dans la case I,J), la variable logique N(I,J) est égale à "Faux" et la variable logique NOT N(I,J) est égale à "Vrai". Dans ce cas un "0" est affiché en colonne 7*I+4 et en ligne 3*J+1, position du chiffre des unités dans la case I, J sinon le programme saute en 3020 pour poursuivre les boucles.

*** 3030
La couleur caractère est fixée au Rouge (1). On notera que la couleur fond de caractère a déjà été fixée au Cyan (6) en 3000. Il est donc inutile de le repréciser.

Cela permettra d'afficher les sommes en Rouge.

*** 3040
Ouverture d'une boucle en I, pour calculer les sommes en colonne (I est l'indice des colonnes, variant de 1 à 3).

*** 3050
a/ Mise à zéro de la variable S dans laquelle on stockera provisoirement la somme des données de la colonne I.

b/ Ouverture d'une boucle en J, variant de 1 à 5, pour recueillir les 5 données de la colonne I.

c/ Sommation dans S des données de la colonne.

d/ fermeture de la boucle en J.

e/ La somme S est placée dans la variable SV(I) (somme "verticale" de la colonne I).

*** 3060
a/ et b/ Place le curseur en colonne 7*I-1, ligne 19 (les sommes sont affichées dans les colonnes 1 à 3 du tableau, sur la ligne 6 du tableau).

La somme d'une colonne peut avoir 6 chiffres (la somme maximale est égale à 99999 x 5 = 499995). Il faut donc l'afficher par une instruction PRINT USING avec une image

comportant 6 caractères dièses (voir la définition de l'image T$ à l'étiquette 40) et placer le chiffre des unité en colonne écran 11 pour la colonne tableau 1, ce qui impose de placer le curseur en colonne 11 - 5 = 6 et justifie la formule donnant le numéro de colonne en fonction de I (rappelons que les colonnes tableau ont 7 caractères de large.

c/ Ferme la boucle en I.

*** 3070 - 3090
Instructions analogues aux instructions 3040 à 3060, pour les sommations par ligne.

Les sommes "horizontales" SH(J) sont affichées dans la colonne 4 du tableau, sur les lignes tableau 1 à 5, donc les lignes écran 3*J+1.

*** 3200 - 3210
Sommation des sommes "verticales" SV(I), pour obtenir la somme totale, notée S.

On remarquera qu'il est inutile de faire la sommation des sommes "horizontales" SH(J), pour vérifier comme le ferait un comptable que l'on obtient le même résultat.

L'ordinateur ne se trompe jamais dans une sommation du moins lorsque l'on a correctement programmé cette sommation!

La somme totale peut comporter jusqu'à 7 chiffres (la valeur maximale est 99999 x 15 = 14999985).
Aussi est elle affichée en colonne 4, ligne 6 du tableau, par une instruction PRINT USING utilisant une image S$ comportant 7 caractères dièse (voir la définition de S$ à l'étiquette 40).

Les sommes SH(I) sont affichées avec six chiffres (T$) et un curseur en colonne 28.

La somme S est affichée avec 7 chiffres (S$) et donc avec un curseur en colonne 27, afin d'aligner le chiffre des unité sur les chiffres des unités des sommes "horizontales".

*** 3300
La couleur des caractères est fixée au Noir (0), pour l'affichage des pourcentages.

*** 3310
Ouverture d'une boucle en I pour calculer les pourcentages "verticaux" (rapport de la somme d'une colonne I à la somme totale, multiplié par 100).

*** 3320
a/ Calcul du pourcentage stocké provisoirement dans la variable P.

b/ Les pourcentages seront affichés avec une décimale, ce qui conduit à 4 caractères (deux pour la partie entière, un pour la décimale, un pour le point décimal), sauf dans le cas où P est égal à 100. Dans ce dernier cas nous afficherons la chaîne "Cent" qui a également 4 caractères. Cela conduit à placer le curseur en colonne écran 7 pour la colonne 1 du tableau, de façon à aligner la décimale sur le chiffre des unités des sommes. Enfin les pourcentages sont affichés sur la ligne 7 du tableau donc sur la ligne 22 écran.

c/ Si P est égal à 100, la chaîne de caractères "Cent" est affichée en Rouge (1), la couleur est remise au Noir (0) et le programme branché en 3340 pour sauter l'affichage de la valeur numérique.

*** 3330
Affichage de la valeur numérique P.

L'image P$, utilisée dans l'instruction PRINT USING comporte deux caractères dièse, pour les deux chiffres de la partie entière, un point, pour le point décimal et un dernier dièse pour la décimale (voir la définition de P$ à l'étiquette 40).

*** 3340
a/ P est stocké dans la variable PV(I). Cette variable ne sera utilisée que dans le cas où nous désirons recopier le tableau sur l'imprimante.

b/ Fermeture de la boucle en I. Une seule boucle étant en service, il est inutile de préciser le nom de la variable après l'instruction NEXT.

*** 3400 - 3430
Instructions analogues aux instructions 3310 - 3340, permettant de calculer et d'afficher les pourcentages "horizontaux", stockés dans les variables PH(J).

Le tableau complet vient d'être affiché avec les sommes et les pourcentages.

Il faut maintenant proposer un "menu" à l'opérateur, lui permettant de choisir entre l'impression du tableau sur l'imprimante, le calcul d'un autre tableau ou la clôture.

Nous ne pouvons afficher ce menu directement, car nous manquons de place disponible. Nous ne pouvons pas non plus effacer l'écran, dès l'affichage du dernier pourcentage, car nous ne laisserions pas à l'opérateur le loisir d'examiner le résultat.

Nous n'autoriserons l'affichage du menu que sur commande de l'opérateur, par exemple sur la frappe de la touche RAZ.

*** 3500

Ces deux instructions remplissent en Bleu caractère (4), les trois cases inutilisées en bas et à droite du tableau.

*** 3510

Affiche le message "Tapez RAZ", en Rouge (1) sur fond Blanc(7) au centre de la case en bas à droite du tableau.

*** 3520

a/ Le programme se met en attente de la frappe d'une touche et place le caractère correspondant dans la variable chaîne de caractères R$.

b/ Si la touche tapée n'est pas la touche RAZ (numéro de code ASCII 12), le programme est rebranché en début d'instruction en attente d'une nouvelle frappe.

Ainsi le programme ne peut franchir l'étiquette 3520 que si la touche RAZ a été frappée.

*** 3530 - 3570

Affichage du menu, après effacement de l'écran, proposant l'impression, un autre tableau ou la clôture.

On notera que les symboles I, T, C utilisés pour la réponse au menu sont affichés en Blanc (7) sur fond Rouge (1), pour mieux se distinguer.

*** 3580
Permet de brancher le programme en :

- 4000 pour l'impression du tableau (la réponse R$ est "I" et le paramètre de branchement de l'instruction ON... GOTO est égal à 2),

- 20 pour refaire un tableau (l'instruction CLEAR en 20 remet à zéro toutes les variables) (la réponse R$ est "T" et le paramètre de branchement est égal à 3),

- 5000 pour clôturer (la réponse R$ est "C" et le paramètre de branchement est égal à 4).

Enfin si la réponse est ni I, ni T, ni C, le paramètre de branchement est égal à 1 et le programme retourne en 3580 pour une nouvelle réponse.

*** 5000
Instructions de clôture effaçant l'écran. Cette instruction étant la dernière du listing, il n'est pas nécessaire de mettre une instruction END.

* * * * * *

Il nous reste maintenant à écrire le programme d'impression du tableau.

L'imprimante thermique PR 90 040, travaillant sur 40 colonnes, est bien adaptée à la reproduction de l'écran.

Nous aurions pu utiliser l'instruction SCREENPRINT, mais cela aurait nécessité de réafficher le tableau sur l'écran avant de le recopier sur l'imprimante (n'oublions pas que nous l'avons effacé pour proposer le menu).

Il est tout aussi simple de le tracer directement par programme, ce qui nous donnera l'occasion d'utiliser les instructions d'impression.

Le programme d'impression se présente de la façon suivante :

```
4000 OPEN "O",1,"LPRT:(40)"

4010 PRINT#1: PRINT#1:GOSUB 4900

4020 PRINT#1 "        A      B      C     TOTAL    %":
     GOSUB 4900

4030 RESTORE 210

4040 FOR J=1 TO 5

4050 READ Z$:PRINT#1," ";Z$;:PRINT#1,USING IP$;
     Z(1,J),Z(2,J),Z(3,J),SH(J),PH(J):GOSUB 4900

4060 NEXT

4070 PRINT#1," TOT";:
     PRINT#1,USING IP$;SV(1),SV(2),SV(3),S:
     GOSUB 4900

4080 PRINT#1,"  %     ";:
     PRINT#1,USING PP$;PV(1),PV(2),PV(3):
     GOSUB 4900

4090 PRINT#1:PRINT#1:PRINT#1:PRINT#1:

4100 CLOSE:GOTO 3530


4900 FOR I=1 TO 40:PRINT#1,"-";:NEXT:
     PRINT#1:RETURN
```

*** 4000

Ouverture du canal 1, dans le sens de la mémoire centrale vers un périphérique (O signifie "OUTPUT" pour sortie vers le périphérique).

Le périphérique est l'imprimante (caractérisé par "LPRT:"). Le nombre de caractères par ligne est fixé à 40.

Dans ce cas particulier, on peut ne pas écrire (40) derrière les deux points, car 40 est la valeur retenue par défaut -(M.Ref 177)-

*** 4010

a/ et b/ Les deux instruction PRINT suivies du symbole dièse et du numéro de canal -(M.Ref 179)-, provoquent deux sauts de ligne sur l'imprimante.

Il faut prendre garde à ne pas oublier le symbole dièse et le numéro de canal. En l'absence du symbole dièse, l'impression se fait sur l'écran et non sur l'imprimante.

c/ Branchement au sous-programme 4900 qui affiche une ligne de 40 tirets (voir 4900).

*** 4020

Affichage de la ligne 0 du tableau, donnant la signification des colonnes et branchement au sous-programme 4900 pour tracer une nouvelle ligne de tirets.

*** 4030

Nous allons utiliser les DATA de l'étiquette 210, pour afficher les symboles dans la colonne 0 du tableau, caractérisant les lignes (jours de la semaine, TOT et %).

Comme ces DATA ont déjà été lues en 220 par le programme principal de construction du tableau à l'écran, le pointeur de DATA se trouve après la denière donnée. Il est donc nécessaire de ramener ce pointeur à la première donnée. C'est l'objet de l'instruction RESTORE 210.

Nous avons déjà remarqué qu'une instruction analogue n'est pas nécessaire dans le programme principal, bien qu'une nouvelle écriture du tableau puisse avoir lieu. En effet la reprise du programme se fait en 20, où l'instruction CLEAR remet à zéro toutes les variables et en particulier le pointeur de DATA.

*** 4040 - 4060

Boucle en J affichant tout d'abord la donnée lue dans les DATA de l'étiquette 210, puis les données Z(1,J) à Z(3,J) de la ligne tableau J, ainsi que la somme SH(J) et le pourcentage PH(J).

On notera la construction de l'instruction PRINT USING, avec le symbole dièse, le numéro de canal (ici 1) et une virgule entre PRINT et USING-(M.Ref 179)-.

L'image utilisée IP$ (définie en 40), comporte les symboles dièse, séparés par des espaces, pour placer les valeurs affichées dans la liste dans une position correcte sur la ligne.

On notera également que la première variable de la liste est précédée par un point-virgule ";", alors que les autres variables de la liste sont séparées par des virgules ",".

Le branchement au sous-programme 4900 affiche des lignes de tirets entre les lignes du tableau.

*** 4070 - 4080
Affichages, en ligne 6 et 7 du tableau, des sommes "verticales", de la somme totale et des pourcentages "verticaux".

L'appel au sous-programme 4900 sépare les lignes par une ligne de tirets.

On notera la construction des deux images utilisées par les instructions PRINT USING, différentes, pour tenir compte des caractéristiques particulières des variables affichées.

*** 4090
Provoque 4 sauts de ligne, permettant de "sortir" le papier de l'imprimante en dégageant le tableau imprimé.

*** 4100
a/ L'impression étant terminée, le canal de transmission des données peut être fermé. C'est l'objet de l'instruction CLOSE -(M.Ref 170)-qui ferme tous les canaux ouverts.

Si l'on avait voulu fermer uniquement le canal 1, il aurait fallu écrire CLOSE 1.

Il est indispensable de fermer les canaux ouverts, sous peine de troubles de fonctionnement ultérieurs.

En particulier il serait impossible d'obtenir un listing sur l'imprimante. Sur la commande LIST "LPRT:", un message d'erreur IU (périphérique en cours de fonctionnement, "device In Use") serait envoyé, signalant que l'imprimante n'est pas disponible.

Pour retrouver l'usage de l'imprimante dans un tel cas, il suffit de frapper CLOSE directement au clavier.

On notera que la fermeture de tous les canaux peut être obtenue également par les instructions END et NEW.

b/ Branchement en 3530 pour l'affichage du menu.

*** 4900
Boucle en I permettant d'afficher 40 tirets "-" sur l'imprimante. On notera le point-virgule derrière "-", pour éviter le saut à la ligne. Le dernier PRINT exécute un saut de ligne pour séparer la ligne de tirets de la ligne imprimée suivante.

Le résultat de l'impression est donné ci-dessous sur un exemple. On constatera qu'il n'est pas possible de tracer des séparations verticales entre les colonnes, car nous ne disposons pas de ce type de caractère (le caractère de numéro de code ASCII 92 est composé de deux tirets verticaux, mais son utilisation n'ajoute pas de clarté au graphisme). Nous laissons au lecteur le soin de l'essayer.

	A	B	C	TOTAL	%
LUN	6790	11570	980	19340	21.8
MAR	5780	8680	1040	15500	17.5
MER	4980	9890	1200	16070	18.2
JEU	5340	10450	1150	16940	19.1
VEN	7840	12050	780	20670	23.4
TOT	30730	52640	5150	88520	
%	34.7	59.5	5.8		

L'imprimante PR 90 080 permet plus de "faintaisies". Il est par exemple possible d'écrire avec des caractères double largeur et de créer des caractères "utilisateur", en particulier tracer des lignes et des colonnes.

Nous allons voir comment programmer ces graphismes spéciaux, dans le cas particulier de notre tableau, mais nous renvoyons le lecteur à l'Annexe II pour plus de détails sur la programmation de cette imprimante.

Une fois le choix d'impression fait par l'opérateur, on lui demandera le numéro sous lequel il veut référencer son tableau (ce numéro sera reproduit en tête du tableau), ainsi que la date (qui sera également imprimée en tête du tableau).

Le tableau sera inscrit dans un cadre composé de lignes de deux points d'épaisseur.

Les séparations verticales entre la première colonne tableau (renseignements lignes, jours de la semaine, total et %) et la seconde, entre la quatrième et la cinquième seront également des lignes de deux points d'épaisseur.

Les lignes de séparation horizontales seront des lignes simples, sauf la première et les deux dernières qui seront constituées de deux lignes séparées.

Nous allons tout d'abord placer des instructions complémentaires en début de programme pour créer les données nécessaires aux graphismes spéciaux.

```
45 IS$= " "+T$+" "+T$+" "+T$+" "+S$

60 MG$=CHR$(8):MC$=CHR$(15):PC$=CHR$(16):
   PGR$=CHR$(27)+PC$:RR$=CHR$(28):
   DL$=CHR$(14):SL$=CHR$(10)


70 P$=CHR$(136):
   P2$=CHR$(140):P3$=CHR$(152):P4$=CHR$(148):
   V$=CHR$(255):VB$=CHR$(252):VH$=CHR$(159):

72 DATA 193,227,213,201,193,227
```

```
75 SIG$="":FOR K=1 TO 6:READ Z:
   SIG$=SIG$+CHR$(Z):NEXT

80 RPA$=RR$+CHR$(23)+P$:RPB$=RR$+CHR$(41)+P$
   RPC$=RR$+CHR$(49)+P$:RPD$=RR$+CHR$(33)+P$

82 RPA2$=RR$+CHR$(23)+P2$:RPB2$=RR$+CHR$(41)+P2$
   RPC2$=RR$+CHR$(49)+P2$:RPD2$=RR$+CHR$(33)+P2$

84 RPA3$=RR$+CHR$(23)+P3$:RPB3$=RR$+CHR$(41)+P3$
   RPC3$=RR$+CHR$(49)+P3$:RPD3$=RR$+CHR$(33)+P3$

86 RPA4$=RR$+CHR$(23)+P4$:RPB4$=RR$+CHR$(41)+P4$
   RPC4$=RR$+CHR$(49)+P4$:RPD4$=RR$+CHR$(33)+P4$

90 LM$=V$+V$+RPA$+V$+V$+RPB$+V$+RPB$+V$+
   RPB$+V$+V$+RPC$+V$+RPD$+V$+V$

92 LH$=VB$+VB$+RPA2$+VB$+VB$+RPB2$+VB$+RPB2$+VB$+
   RPB2$+VB$+VB$+RPC2$+VB$+RPD2$+VB$+VB$

94 LB$=VH$+VH$+RPA3$+VH$+VH$+RPB3$+VH$+RPB3$+VH$+
   RPB3$+VH$+VH$+RPC3$+VH$+RPD3$+VH$+VH$

96 LN$=V$+V$+RPA4$+V$+V$+RPB4$+V$+RPB4$+V$+
   RPB4$+V$+V$+RPC4$+V$+RPD4$+V$+V$

97 PG$=PGR$+CHR$(0)+CHR$(120)

98 TAB$=PC$+"20"
```

45

IS$ sera utilisée comme image pour imprimer la ligne des sommes verticales et de la somme totale (voir 4070 où sera justifiée l'utilisation de cette image particulière alors que dans le programme précédent d'impression nous avions utilisé la même image que celle employée pour l'impression des lignes de données, c'est-à-dire IP$, définie en 40).

**** 60

Définition des caractères de commande de l'imprimante (voir Annexe II).

Ces caractères sont utilisés pour changer le Mode d'Impression, pour positionner les chaînes, etc.

*** 70
Définitions de divers "motifs".

Le motif P$ est un point situé sur la ligne centrale d'une ligne texte.

Rappelons qu'une ligne texte sur l'imprimante est composée de 7 lignes graphiques, que nous numérotons de 1 pour la ligne supérieure à 7 pour la ligne inférieure. Le "poids" du point 1 est 1, le poids du point N est 2 puissance (N - 1). Ainsi le poids du point central 4 est-il 8, et P$ est défini par le nombre 128 + 8 = 136.

P$ est donc le motif qui nous permettra de bâtir une ligne graphique au centre de la ligne texte.

Le motif P2$ est composé des points 3 et 4. Il permettra de bâtir une ligne d'épaisseur double.

Le motif P3$ est composé des points 4 et 5. Il permettra de bâtir également une ligne d'épaisseur double, mais placée différemment de la ligne précédente par rapport à la ligne centrale.

Le motif P4$ est composé des points 3 et 5. Il permettra de bâtir deux lignes situées de part et d'autre de la ligne centrale.

Le motif V$ est composé des 7 points. C'est le motif de séparation verticale.

Le motif V8$ est composé des points 3, 4, 5, 6 et 7. Il sera utilisé pour compléter la ligne constituée des motifs P2$, par des séparations verticales situées sous la ligne.

Enfin VH$ est composé des points 1, 2, 3, 4 et 5. Il sera utilisé pour compléter la ligne constituée des motifs P3$, par des séparations verticales situées au dessus de la ligne.

*** 72 - 75
Les instructions sous les deux étiquettes définissent le caractère spécial SIG$ représentant le caractère grec Sigma majuscule. Ce caractère est l'assemblage de six motifs dont les valeurs caractéristiques sont stockées dans les DATA de 72.

*** 80 - 86

Le caractère RPA$ est constitué de la répétition de 23 motifs P$ (voir l'utilisation de la commande de répétition RR$ en annexe II). Il constitue donc un trait de 23 points de long, sur la ligne centrale d'une ligne texte.

Les caractères RPB$, RPC$, RPO$ sont analogues mais de longueur différentes, 41, 49 et 33 points.

Les caractères RPA2$, RPB2$, ...RPD4$, sont construits de façon analogue à partir des motifs P2$, P3$ et P4$. Ils constituent des éléments de lignes doubles de différentes longueurs.

*** 90 - 96

Le caractère spécial LM$ résulte de la concaténation des caractères V$, RPA$, RPB$, RPC$ et RPO$. Il constitue en définitive une ligne, au centre de la ligne texte, de 2 + 23 + 2 + 2 x (41 + 1) + 41 + 2 + 49 + 1 + 33 + 2 = 239 points, avec, intercalées des séparations verticales de 7 points de haut.

Ce caractère sera utilisé pour la séparation des lignes de données.

Les caractères LH$ (ligne supérieure du cadre), LB$ (ligne inférieure du cadre), LN$ (deux dernières lignes de séparation pour les sommes et les pourcentages verticaux), sont bâtis de façon analogue.

*** 97

PG$ est un caractère de positionnement en Mode Graphique, plaçant le caractère graphique suivant au point 120 (en numérotant 0 le premier point sur la marge gauche).

*** 98

TAB$ est un caractère de positionnement en Mode Caractère, plaçant le caractère suivant en colonne 20 (En numérotant 0 la première colonne).

Il nous faut prévoir quelques instructions permettant à l'opérateur d'entrer le numéro du tableau et la date. Nous placerons ces instructions à partir de l'étiquette 3900, ce qui nous oblige à changer l'étiquette de branchement du ON .. GOTO en 3590, lorsque l'opérateur a choisi l'impression.

```
3590 ON 1-(R$="I")-2*(R$="T")-3*(R$="C")
     GOTO 3580,3900,20,5000

3900 CLS:LOCATE 5,10:PRINT "Numero du tableau  :":
     LOCATE 5,14:PRINT "Date (JJ/MM/AA) :"

3910 LOCATE 25,10:INPUT NN$:
     LOCATE 25,14:INPUT DD$

3920 CLS:LOCATE 5,10:
     PRINT "IMPRESSION DU TABLEAU NUMERO ";NN$

3930 LOCATE 16,14:PRINT DD$
```

*** 3590
Branche, entre autres, à l'étiquette 3900, si l'opérateur a tapé la touche I, pour commander l'impression du tableau.

*** 3900
Affiche deux messages à l'écran pour demander le numéro du tableau et la date (on précise la façon d'entrer la date, pour normaliser la présentation sur l'imprimante).

*** 3910
Le numéro est stocké dans la variable chaine NN$ et la date dans la variable chaine DD$.

*** 3920
Affichage d'un message à l'écran rappelant à l'opérateur le numéro et la date.

Le programme d'impression proprement dit, commence à l'étiquette 4000.

Il comprendra successivement des commandes d'impression en Mode Caractère Double (par DL$), des commandes d'impression en Mode Caractère Simple (par MC$) et des commandes d'impression en Mode Graphique (par MG$).

Il est nécessaire d'avoir lu l'Annexe II, pour bien comprendre la syntaxe de ces instructions.

```
4000 OPEN "O",1,"LPRT:(80)"

4010 PRINT#1,DL$;PC$;"30";"TABLEAU ";NN$;SL$;
     SL$;PC$;"32";DD$;SL$

4020 PRINT#1,MG$;PG$;LH$

4030 GOSUB 4900:PRINT#1,MC$;TAB$;
     "        A       B       C      TOTAL    %";:
     GOSUB 4900:GOSUB 4900:GOSUB 4920

4040 RESTORE 210

4050 FOR J=1 TO 5

4060 GOSUB 4900:READ Z$:PRINT#1,MC$;TAB$;" ";Z$;:
     PRINT#1,USING IP$;Z(1,J),Z(2,J),Z(3,J),
     SH(J),PH(J);:
     GOSUB 4900:GOSUB 4900:
     IF J=5 THEN GOSUB 4920 ELSE GOSUB 4910

4070 NEXT

4080 GOSUB 4900:PRINT#1,MG$;PGR$;CHR$(0);
     CHR$(132);SIG$;

4090 PRINT#1,MC$;PC$;"24";:
     PRINT#1,USING IS$;SV(1),SV(2),SV(3),S;:
     GOSUB 4900:GOSUB 4900:GOSUB 4920

4100 GOSUB 4900:PRINT#1,MC$;TAB$;"  %     ";:
     PRINT#1,USING PP$;PV(1),PV(2),PV(3);:
     GOSUB 4900:GOSUB 4900

4110 PRINT#1,MG$;PG$;LB$

4120 FOR K=1 TO 5:PRINT#1:NEXT

4130 PRINT#1,MC$:CLOSE 1:GOTO 3530


4900 PRINT#1,MG$;PG$:V$;V$;PGR$;CHR$(0);CHR$(145);
     V$;V$;PGR$;CHR$(0);CHR$(188);V$;PGR$;CHR$(0);
     CHR$(230);V$;PGR$;CHR$(1);CHR$(16);V$;V$;
     PGR$;CHR$(1);CHR$(67);V$;PGR$;CHR$(1);
     CHR$(101);V$;V$:RETURN
```

```
4910 PRINT#1,MG$;PG$;LM$:RETURN

4920 PRINT#1,MG$;PG$;LN$:RETURN
```

*** 4000

Ouverture du canal 1 pour alimenter, en sortie de mémoire centrale ("O" pour Output), le périphérique imprimante ("LPRT:(80)"), travaillant sur 80 colonnes.

*** 4010

Impression d'une succession de commandes faisant exécutar les opérations suivantes :

- Commande du Mode Caractère Oouble (OL$).
- Positionnement de la tête d'impression en colonne texte numéro 30 (c'est à dire en laissant 30 espaces).
- Impression de la chaine de caractères "TABLEAU ", en taille double.
- Impression de la chaine NN$, représentant le numéro du tableau, à la suite de la chaine précédente (le point-virgule interdit le saut à la ligne).
- Exécution de deux sauts à la ligne (SL$).
- Positionnement de la tête d'impression en colonne texte numéro 32 (c'est à dire en laissant 32 espaces).
- Impression de la chaine DO$, représentant la date.
- Exécution d'un saut à la ligne (SL$).

*** 402D

Impression, en Mode Graphique (MG$), de la ligne supérieure LH$ du tableau, le premier point se trouvant placé au point 120 (PG$ défini en 97).

*** 4030

a/ Branchement sur le sous-programme 4900.

Le sous-programme 4900 place des traits verticaux de séparation V$, de façon à bâtir les lignes verticales de séparation des colonnes du tableau.

Chaque trait vertical est positionné par une commande de la forme PG$;CHR$(N1);CHR$(N2) (voir l'Annexe II, pour le calcul de N1 et N2 en fonction de la position N du caractère à placer).

b/ Passage en Mode Caractère, positionnement en colonne 20, impression de la ligne de renseignement des colonnes du tableau.

Le point-virgule en fin d'instruction évite un saut à la ligne.

c/ L'appel au sous-programme 4900 imprime les séparations verticales sur la ligne précédemment imprimée.

d/ Ce deuxième appel au sous-programme 4900 imprime les séparations verticales sur une ligne vide comme l'instruction a. Cela permet d'aérer la présentation.

e/ Appel au sous-programme 4920. Ce sous-programme affiche la ligne de séparation horizontale double LN$. Il faut noter que 4920 comme 4900 fait passer en Mode Graphique. Il est donc nécessaire de repasser en Mode Caractère par MC$ avant une impression de caractère.

*** 4040

Comme dans le programme d'impression avec l'imprimante PR 90 040, il est nécessaire de replacer le pointeur de DATA en début des données stockées sous l'étiquette 210, pour pouvoir les relire

*** 4050 - 4070

Boucle en J, destinée à afficher les cinq lignes des données ainsi que les sommes et pourcentages horizontaux.

a/ L'appel au sous programme 4900 provoque l'impression d'une première ligne de traits verticaux de séparation.

b/ Lecture de la chaine Z$ dans la liste de DATA en 210

c/ Passage au Mode Caractère (MC$), tabulation de 30 espaces, impression d'un espace puis de Z$ (jour de la semaine).

Le point-virgule en fin d'instruction évite le saut de ligne.

d/ Impression avec l'image IP$ (voir 40) des trois données de la ligne J du tableau, de la somme et du pourcentage. Là encore le point-virgule en fin d'instruction évite le saut à la ligne.

e/ L'appel au sous-programme 4900 imprime les traits verticaux de séparation sur la ligne où viennent d'être imprimées les données.

f/ Impression des traits verticaux de séparation sur la ligne vide suivante.

g/ Impression de la ligne simple de séparation horizontale LM$ (sous-programme 4910) sauf pour la dernière ligne du tableau (J = 5), pour laquelle on imprime la ligne double LN$ (sous-programme 4920).

*** 4090 - 4100
Ces deux instructions impriment de façon analogue aux instructions précédentes, les deux dernières lignes du tableau.

Dans le programme utilisé pour l'imprimante thermique PR 90 040, nous avions utilisé l'image IP$ pour imprimer la ligne affichant les sommes verticales et la somme totale. L'absence de cinquième valeur à imprimer dans la liste des variables figurant dans le PRINT USING (il n'y a pas de pourcentage à afficher sur cette ligne) était sans importance.
Mais il faut savoir que la non utilisation de la totalité de l'image provoque automatiquement un saut à la ligne, même si l'instruction se termine par un point-virgule.

L'utilisation de IP$ nous empêcherait donc d'imprimer les traits verticaux de séparation sur cette ligne du tableau, puisque ces traits sont affichés après l'impression des données en évitant le saut à la ligne.

Nous sommes donc amenés à utiliser une image IS$ correspondant à l'impression des quatre sommes et uniquement des quatre sommes (IS$ a été définie en 45).

*** 4110
Impression de la ligne inférieure du cadre LB$.

*** 4120
Boucle en K permettant d'exécuter 5 sauts à la ligne, dégageant le tableau de l'imprimante.

*** 4130
a/ Commande du Mode Caractère pour remettre l'imprimante en état de travail normal.

Son oubli interdirait un listing ultérieur, car la machine ne peut imprimer un listing en Mode Caractère si elle est en Mode Graphique.Si cet oubli est commis, il suffit de couper puis de rétablir l'alimentation de l'imprimante.

b/ Fermeture du canal 1.
Cette commande est également indispensable. Les remarques faites pour l'imprimante thermique, sont valables pour l'imprimante à aiguilles (c'est un problème de canal, non de périphérique).

c/ Branchement en 3530 pour proposer un nouveau choix à l'opérateur (on peut éditer plusieurs copies du tableau).

Nous donnons ci-dessous une reproduction d'un exemple de tableau.

TABLEAU 52

20/09/83

	A	B	C	TOTAL	%
LUN	6790	11570	980	19340	21.8
MAR	5780	8680	1040	15500	17.5
MER	4980	9890	1200	16070	18.2
JEU	5340	10450	1150	16940	19.1
VEN	7840	12050	780	20670	23.4
Σ	30730	52640	5150	88520	
%	34.7	59.5	5.8		

10

Sauvegarde des données

Il est très souvent nécessaire de conserver des données et des résultats pour une utilisation ultérieure. Cette sauvegarde peut se faire sur la cassette, sous forme d'un fichier de données et quelque fois sous forme d'un fichier binaire.

Nous allons voir des exemples de ces deux types de sauvegarde, ce qui nous permettra d'en connaître les avantages et les limitations.

Le programme de fabrication de caractères utilisateur se terminait -(M.Bas 63)- par l'impression des suites de huit nombres définissant chaque caractère. Il peut être commode, lorsque l'on a créé un grand nombre de caractères de ne pas avoir à recopier cette liste de données au début du programme d'utilisation, comme nous l'avons fait pour le programme de mini-écriture (-(M.Bas 127)-.

Nous allons donc modifier le programme de création de caractères utilisateur, de façon à pouvoir en stocker les résultats.

La première méthode, très générale consiste à créer un fichier de données sur la cassette et à relire ce fichier en début de programme d'utilisation.

Nous modifierons l'instruction 3000 (-(M.Bas 100)-, de façon à proposer à l'opérateur un menu comportant un choix supplémentaire, la sauvegarde des résultats.

Les instructions 3000 à 5000 du programme restent les mêmes à l'exception des suivantes :

```
3000 LOCATE 0,12:
     PRINT "POUR IMPRIMER LES RESULTATS TAPEZ ";:
     COLOR 0,1:PRINT "ENTREE":COLOR 4,6

3002 LOCATE 0,14:
     PRINT "POUR LES SAUVEGARDER          TAPEZ ";:
     COLOR 0,1:PRINT "S":COLOR 4,6

3004 LOCATE 0,16:
     PRINT "POUR CLOTURER                 TAPEZ ";:
     COLOR 0,1:PRINT "F":COLOR 4,6

3010 LOCATE 20,16:R$=INPUT$(1)

3020 ON 1-(ASC(R$)=13)-2*(R$="F")-3*(R$="S")
     GOTO 3010,3500,4000,6000

3500 CLS:LOCATE 8,12:
     PRINT "IMPRESSION DES RESULTATS":
     OPEN "O",1,"LPRT:(80)"
```

Il est inutile de commenter en détail ces instructions de routine. Les instructions 3000 - 3004 affichent le menu. Les instructions 3010 - 3020 effectuent le branchement, en 3500 pour l'impression, en 6000 pour le stockage, en 5000 pour la clôture et en 3010 en cas d'erreur de frappe.

Nous avons rajouté quelques instructions en 3500, avant l'ouverture du canal d'impression, pour effacer l'écran et afficher un message précisant à l'opérateur que l'on est en phase d'impression. Cela permet de noter plus clairement le retour au menu, une fois cette phase terminée.

Le programme de stockage des résultats est le suivant :

```
6000 CLS:LOCATE 7,12:
     PRINT "NOM DU FICHIER DE STOCKAGE"

6010 LOCATE 15,16:INPUT F$
```

```
6020 BEEP:CLS:LOCATE 5,12:
     PRINT "METTRE LES TOUCHES DU LECTEUR":
     LOCATE 7,14:PRINT "EN POSITION ";:COLOR 1:
     PRINT "ENREGISTREMENT":COLOR 4

6030 LOCATE 20,22:
     PRINT "TAPEZ ";:COLOR 0,3:PRINT "ENTREE":
     COLOR 4,6:BEEP

6040 R$=INPUT$(1):IF ASC (R$)<>13 GOTO 6040

6500 CLS:OPEN "O",1,F$

6510 PRINT #1,NG:LOCATE 6,0:
     PRINT "SAUVEGARDE DES ";NG+1;" CARACTERES"

6520 FOR I=0 TO NG

6530 COLOR 1:PRINT I;:COLOR 4

6540 FOR J=1 TO 8

6550 PRINT #1,Z(J,I):PRINT Z(J,I)

6560 NEXT J,I

6570 CLOSE:GOTO 3000
```

*** 6000 - 6010

Ces deux instructions permettent à l'opérateur d'entrer le nom du fichier dans lequel il désire stocker ses données.

Ce nom est placé dans la variable chaîne de caractères F$.

*** 6020 - 6040

Affichage d'un message à l'opérateur pour lui rappeler que le lecteur de cassette doit être placé en situation d'enregistrement, d'où l'usage du BIP et de la couleur Rouge pour attirer son attention.

L'enregitrement ne peut en outre s'effectuer que sur commande de la touche ENTREE.

*** 6500

a/ Effacement de l'écran.

b/ Ouverture du canal 1, fonctionnant dans le sens "sortie" (O pour Output), c'est-à-dire dans le sens mémoire centrale vers le périphérique, ici le lecteur de cassette, qui est le périphérique choisi par défaut lorsque le type de périphérique n'est pas précisé -(M.Ref 178)-.

On notera que le nom du fichier F$ est précisé à la fin de l'instruction.

Une fois le programme exécuté, le fichier figure sur la cassette sous le nom chargé dans F$, suivi du suffixe .DAT, précisant qu'il s'agit d'un fichier de données et non d'un programme écrit en BASIC dont le suffixe est .BAS.

*** 6510

a/ Charge dans le fichier en première position la valeur NG, c'est-à-dire le nombre de caractères créés (plus précisément le nombre de caractères moins un, puisque le premier caractère porte le numéro 0).

C'est la présence du caractère dièse suivi du numéro de canal qui fait que l'instruction PRINT charge NG dans le fichier et ne l'affiche pas à l'écran.

b/ et c/ Affiche un message à l'écran comportant la valeur NG+1.

C'est l'absence du caractère dièse à la suite de l'instruction PRINT qui permet l'affichage à l'écran.

*** 6520 - 6560

Boucles en I (numéro du caractère variant de 0 à NG) et J (variant de 1 à 8, pour balayer les huit nombres caractéristiques du caractère), chargeant les données dans le fichier.

Il faudra prendre garde à utiliser deux boucles emboitées de la même façon pour relire les données.

Les données sont ainsi stockées dans l'ordre suivant :

NG, Z(1,0), Z(2,0), Z(3,0), ..., Z(8,0), Z(1,1), Z(2,1), ..., Z(1,NG), ..., Z(8,NG).

On notera le chargement de Z(J,I) dans le fichier par l'instruction PRINT suivie du caractère dièse et du numéro de canal et l'affichage de la même valeur à l'écran par l'instruction PRINT simple.

*** 6570

a/ Fermeture de tous les canaux -(M.Ref 170)-

Cette instruction est indispensable pour pouvoir réouvrir le canal 1 pour une autre utilisation (en particulier l'impression sur l'imprimante) et pour forcer le canal à se "vider" en fin de chargement. Son oubli risque de provoquer la non mise en mémoire dans le fichier des dernières valeurs Z(I,J).

b/ Retour au menu.

Il aurait été possible, en utilisant les boucles en I et J, de simultanément stocker les résultats dans le fichier, les afficher à l'écran et les imprimer. Il aurait été nécessaire, pour cela, d'ouvrir un premier canal, numéro 1 par exemple, pour l'imprimante (instruction 3500) et un second, numéro 2, pour le fichier (instruction 6500 dans laquelle le numéro de canal est posé égal à 2).

L'orientation d'une donnée vers l'imprimante se fait alors par l'instruction PRINT # 1, l'orientation vers le fichier par l'instruction PRINT # 2 et l'orientation vers l'écran par l'instruction PRINT simple.

* * * * * *

Il nous faut maintenant donner un exemple de programme utilisant les données du fichier et en conséquence relisant le fichier pour pouvoir bâtir les caractères utilisateur.

Le programme proposé pourra être utilisé pour vérifier que les données stockées sont correctes et correspondent bien à l'utilisation prévue.

```
10 CLS:LOCATE 7,12:
   PRINT "NOM DU FICHIER DE STOCKAGE"

20 LOCATE 15,16:INPUT F$:CLS

30 OPEN "I",1,F$

40 INPUT #1,NG:LOCATE 6,0:
   PRINT "NOMBRE DE CARACTERES STOCKES ";NG+1

50 POKE 24575,NG

60 CLEAR ,,NG+1

70 NG=PEEK(24575)

80 DIM Z(8,NG)

100 FOR I=0 TO NG

110 COLOR 1:PRINT I;:COLOR 4

120 FOR J=1 TO 8

130 INPUT #1,Z(J,I):PRINT Z(J,I)

140 NEXT J:PRINT:NEXT I

150 CLOSE

160 PRINT:PRINT "         TAPEZ ";:
    COLOR 0,3:PRINT "ENTREE":COLOR 4,6

170 R$=INPUT$(1):IF ASC(R$)<>13 GOTO 170


200 CLS:FOR I=0 TO NG

210 DEFGR$(I)=Z(1,I),Z(2,I),Z(3,I),Z(4,I),
              Z(5,I),Z(6,I),Z(7,I),Z(8,I)

220 PRINT I,GR$(I)

230 NEXT
```

*** 10 - 20
Lecture du nom du fichier de stockage, tapé au clavier par l'opérateur.

*** 30
Ouverture du canal 1 en "entrée" (I pour Input), c'est-à-dire dans le sens périphérique vers la mémoire centrale. Le périphérique n'étant pas précisé, c'est le lecteur de cassette qui est retenu par défaut -(M.Ref 178)-. On notera que le nom du fichier F$ est placé en fin d'instruction.

Si le lecteur de cassette est placé en situation de lecture, le moteur est mis en marche et la cassette se déroule jusqu'à ce que le fichier, dont le nom est stocké dans F$, soit rencontré. Il faut donc prendre garde à positionner correctement la bande de façon à se placer avant l'enregistrement du fichier.

*** 40
a/ Lecture dans le fichier de la première valeur stockée, c'est-à-dire le nombre NG.

L'instruction INPUT recueille la donnée dans le fichier et non au clavier car elle est suivie du caractère dièse et du numéro de canal.

b/ et c/ Affichage à l'écran du nombre NG+1.

*** 50 - 60
L'instruction CLEAR efface toutes les valeurs en mémoire centrale après avoir effectué les réservations, en particulier, dans notre cas, la réservation des NG+1 places pour les caractères utilisateur.

Mais dans ces conditions la valeur NG est détruite !

Pour la sauver de l'effacement, nous plaçons, par l'instruction 50, sa valeur dans la mémoire 24575.

Il se trouve en effet que les 192 mémoires d'adresses comprises entre 24384 (5F40 en hexadécimal) et 24575 (5FFF) se trouvent disponibles.

On prendra garde à ce que ces mémoires sont disponibles sur la version actuelle du BASIC TO7. Il se peut que des versions ultérieures utilisent ces emplacements mémoire.

L'instruction POKE M,X place la valeur X dans l'octet d'adresse M -(M.Ref 86)-. Bien entendu la valeur maximale de X est 255 et la valeur minimale 0. C'est bien notre cas puisque le nombre de caractères utilisateur ne peut dépasser 128.

L'instruction CLEAR efface toutes les variables, mais n'efface pas les mémoires. La mémoire 24575 contient donc encore la valeur NG après l'exécution de l'instruction 60. En réalité l'instruction CLEAR n'efface pas les mémoires où sont stockées les variables, pas plus qu'elle n'efface la mémoire 24575. Elle ne fait que rendre disponibles les mémoires de stockage des variables pour une nouvelle utilisation.

On notera que le nom du fichier F$ est effacé par l'instruction 60, mais cela n'a pas d'importance puisque nous n'en avons plus besoin.

*** 70
Récupération de la valeur de NG stockée en 24575.

L'instruction X=PEEK(M) place dans X le contenu de la mémoire de numéro M -(M.Ref 132)-

*** 80
Il est alors possible de dimensionner la variable Z(J,I) destinée à recueillir les données lues dans le fichier.

*** 100 - 150
Boucles en I (de 0 à NG) et en J (de 1 à 8) pour relire les données. On notera que ces boucles sont "calquées" sur les boucles de stockage du programme de création des caractères utilisateur, de façon à relire les données dans l'ordre de leur écriture.

*** 150
Fermeture du canal pour une utilisation ultérieure. Cette instruction est indispensable pour forcer le canal a se "vider" complètement en fin de lecture. L'oubli de cette instruction pourrait provoquer le non chargement des dernières données dans les variables Z(I,J).

*** 160 - 170
Ces deux instructions permettent de bloquer le programme afin de laisser à l'opérateur le loisir d'observer l'affichage des données.

*** 200 - 230
Boucle en I permettant de bâtir les caractères utilisateur GR$(I) et de les afficher pour contrôle.

* * * * * *

Cette méthode de stockage de données est très générale. Elle n'est pas limitée au stockage des variables numériques. On peut stocker des chaînes de caractères de la même façon par les instructions PRINT # 1. Ce n'est qu'à la relecture du fichier que la valeur lue sera chargée dans une variable numérique ou dans une variable chaîne de caractères.

Il faut bien comprendre que les valeurs sont chargées dans le fichier sous forme de chaîne de caractères, une valeur numérique étant représentée par la suite des caractères chiffres qui la compose (ainsi que le point décimal,le signe - et les caractères E ou D précédant éventuellement l'exposant).

A la relecture, l'instruction INPUT #1,Z$ interprètera la valeur lue comme une chaîne et la placera dans la variable chaîne Z$. Par contre l'instruction INPUT #1,Z interprètera la valeur lue comme une variable numérique (si la chaîne de caractères lue est interprètable comme telle), et la placera dans la variable numérique Z.

* * * * * *

Nous remarquerons que nous avons séparé en deux phases nettement distinctes le programme d'utilisation du fichier, une phase de lecture des données avec leur chargement dans la variable Z(J,I) et une phase de définition des caractères utilisateur GR$(I). Cette méthode est plus claire mais elle a l'inconvénient de charger inutilement la mémoire centrale avec le tableau des Z(J,I).

Nous proposons donc la version améliorée suivante du programme d'utilisation du fichier (la partie affichage des caractères utilisateur a été également modifiée).

```
10 CLS:LOCATE 7,12:
   PRINT "NOM DU FICHIER DE STOCKAGE"

20 LOCATE 15,16:INPUT F$:CLS

30 OPEN "I",1,F$

40 INPUT #1,NG:POKE 24575,NG

50 CLEAR ,,NG+1

60 NG=PEEK(24575)


100 FOR I=0 TO NG

110 FOR J=1 TO 8:INPUT #1,Z(J):NEXT J

120 DEFGR$(I)=Z(1),Z(2),Z(3),Z(4),
               Z(5),Z(6),Z(7),Z(8)

130 NEXT I

140 CLOSE


200 I=0:A$=CHR$(22):FD$=A$+",":FG$=A$+","

210 CLS:LOCATE 19,5,0:PRINT I

220 COLOR 1:LOCATE 20,10:PRINT GR$(I)

230 COLOR 0:LOCATE 10,15:ATTRB 1,0:PRINT GR$(I)

240 LOCATE 20,15:ATTRB 1,1:PRINT GR$(I)

250 LOCATE 30,15:ATTRB 0,1:PRINT GR$(I):
    COLOR 4:ATTRB 0,0

260 LOCATE 5,18:
    PRINT "CARACTERE PRECEDENT TAPEZ ";FG$

270 LOCATE 5,20:
    PRINT "CARACTERE SUIVANT   TAPEZ ";FD$
```

```
280 LOCATE 5,22:
    PRINT "CLOTURE                TAPEZ ";:
    COLOR 1,3:PRINT "STOP":COLOR 4

290 R$=INPUT$(1):R=ASC(R$)

300 ON 1-(R=8)-2*(R=9)-3*(R=2)
    GOTO 290,500,600,700


500 I=I-1:IF I=-1 THEN I=0

510 GOTO 210


600 I=I+1:IF I=NG+1 THEN I=NG

610 GOTO 210


700 CLS
```

*** 10 - 60

Ces instructions sont identiques, à des détails près, aux instructions 10 à 70 du programme précédent. Seule l'instruction OIM (étiquette 80) n'est pas reprise puisque nous n'utilisons plus la variable Z(J,I).

*** 100 - 200

La boucle en J, à l'intérieur de la boucle en I (de 0 à NG pour la construction des NG+1 caractères utilisateur), permet de recueillir les huit valeurs Z(J) caractéristiques du caractère GR$(I). La variable Z(J) n'a pas été dimensionnée puisque J varie seulement de 1 à 8.

*** 200

a/ Initialisation de la variable I à zéro.

b/ c/ et d/ Construction des caractères Flèche Droite FD$ et Flèche Gauche FG$.

*** 210 - 250

Affichage du numéro I du caractère GR$(I) en ligne 5 écran, du caractère en Rouge en ligne 10 et du caractère en Noir avec les trois tailles doubles en ligne 15.

*** 260 - 280
Affichage de trois messages à l'écran proposant à l'opérateur de lire les caractères suivant ou précédent ou de clôturer.

*** 290 - 300
Branchement aux instructions 500 (caractère précédent, Flèche Gauche R = 8), 600 (caractère suivant, Flèche Droite R = 9), 700 (clôture, R = 2) ou 290 (erreur de frappe).

*** 500
a/ Le caractère précédent ayant été demandé par l'opérateur, I est décrémenté de 1.

b/ Si I prend, après décrémentation la valeur interdite -1, I est remis à zéro.

*** 510
Branchement en 210 pour affichage du nouveau caractère.

*** 600 - 610
Instructions analogues aux précédentes pour passer au caractère suivant (I ne peut dépasser la valeur limite NG).

*** 700
Instruction de clôture effaçant l'écran.

* * * * * *

Nous allons maintenant examiner la deuxième méthode de stockage des données sous forme de fichier binaire.

Il est nécessaire pour effectuer ce type de stockage de ne disposer que de données numériques de valeurs comprises entre 0 et 255 ou de données chaînes de caractères préalablement transformées en suites de codes ASCII (nous verrons ce dernier cas plus loin).

Dans notre cas, les données répondent bien à cette obligation puisque les variables Z(J,I) sont comprises, par construction, entre 0 et 255.

Nous chargerons donc les mémoires "hautes" disponibles avec les valeurs des Z(J,I).

Par mémoires hautes nous entendons les mémoires situées "au-dessous" de la mémoire d'adresse 49151 (BFFF en hexadécimal), pour les machines équipées de la mémoire additionnelle de 16K, ou "au-dessous" de la mémoire d'adresse 32767 (7FFF en hexadécimal), pour les machines sans mémoire additionnelle.

Ce chargement s'effectuera à l'aide de l'instruction POKE, déjà vue plus haut, en faisant décroître les adresses de stockage à partir de l'adresse maximale.

Il est alors nécessaire de "protéger" ces mémoires contre une utilisation intempestive par le programme BASIC.

Les programmes BASIC stockent en effet les variables chaînes de caractères (calculées par concaténation ou par toute fonction comme LEFT$, RIGHT$, MID$, STR$ ou CHR$) à partir de la plus haute mémoire disponible. Pour interdire l'utilisation des mémoires hautes nous utiliserons l'instruction CLEAR ,M -(M.Ref 32)- où M est l'adresse de la mémoire la plus haute laissée à la disposition de l'interprèteur BASIC pour stocker les chaînes.

Nous placerons le nombre NG en 49151, et les valeurs des Z(J,I) en décroissant à partir de 49150. Comme il y a 128 x 8 = 1024 valeurs à stocker au maximum, l'adresse la plus basse de stockage est 48127. Nous réserverons donc ces mémoires par une instruction CLEAR ,48126.

Le programme de création des caractères utilisateur est alors modifié comme suit :

```
10 CLEAR.4000,48126,3

6000 CLS:LOCATE 7,12:
     PRINT "NOM DU FICHIER DE STOCKAGE"

6010 LOCATE 15,16:INPUT F$

6020 BEEP:CLS:LOCATE 5,12:
     PRINT "METTRE LES TOUCHES DU LECTEUR":
     LOCATE 7,14:PRINT "EN POSITION ";:COLOR 1:
     PRINT "ENREGISTREMENT":COLOR 4
```

```
6030 LOCATE 20,22:PRINT "TAPEZ ";:COLOR 0,3:
     PRINT "ENTREE":COLOR 4,6:BEEP

6040 R$=INPUT$(1):IF ASC (R$)<>13 GOTO 6040

6500 CLS:M0!=49151:POKE M0!,NG

6510 LOCATE 6,0:PRINT "SAUVEGARDE DES ";NG+1;:
     " CARACTERES"

6520 FOR I=0 TO NG

6530 COLOR 1:PRINT I:COLOR 4

6540 FOR J=1 TO 8

6550 POKE M0!-8*I-J,Z(J,I):PRINT Z(J,I);

6560 NEXT J:PRINT:NEXT I

6570 SAVEM F$,48127,49151,0

6580 GOTO 3000
```

*** 10

L'instruction 10 a été modifiée de façon à faire la réservation des mémoires hautes au delà de la mémoire 48126.

On notera que la valeur du premier paramètre (4000) de CLEAR, est la réservation des places mémoires pour le calcul des chaînes de caractères, que la valeur du second paramètre (48126), est l'adresse de la plus haute mémoire utilisable par l'interprèteur BASIC et que la valeur du troisième paramètre (3), est la réservation de places pour les trois caractère graphiques utilisés dans le programme.

*** 6000 - 6040

Ces instructions sont les mêmes que dans le programme de stockage développé plus haut.

*** 6500

a/ Effacement de l'écran

b/ La variable M0! stocke l'adresse de la mémoire 49151.

Il est nécesseire de rajouter le symbole ! derrière le nom de variable M0 pour préciser que M0 est une variable simple précision (l'instruction 20 du programme de création des caractères -(M.Bas 68)- impose à toutes les variables d'être des variables entières par DEFINT A-Z). Or 49151 est une valeur trop grande pour être stockée dans une variable entière (valeur maximale 32767 -(M.Ref 14)-).

c/ La mémoire d'adresse 49151 est chargée avec la valeur NG.

*** 6510
Identique à l'instruction 6510 du précédent programme.

*** 6520 - 6560
8oucles en I (variant de 0 à NG) et en J (variant de 1 à 8), pour charger les valeurs des Z(J,I). La valeur Z(J,I) est placée dans la mémoire d'adresse M0!-8*I-J, soit 49150 pour la première valeur Z(1,0) et 48127 pour la dernière valeur possible Z(8,127)

Les valeurs de I sont en outre affichées en Rouge en début de ligne et les huit valeurs des Z(J,I) affichées en Bleu à la suite, de façon à pouvoir suivre le déroulement du programme.

*** 6570
Sauvegarde, dans le fichier binaire dont le nom est dans la variable F$, des mémoires dont les adresses sont comprises entre 48127 et 49151. Le 0 figurant comme dernier paramètre est indispensable (le dernier paramètre est l'adresse d'appel du sous-programme écrit en binaire si la sauvegarde porte sur un programme écrit en langage machine, ce qui n'est pas le cas ici).

On notera que l'instruction de sauvegarde est SAVEM et non SAVE comme pour un programme BASIC.

Une fois le programme exécuté le fichier figure sur la cassette sous le nom stocké dans F$ suivi du suffixe .BIN qui précise que nous avons affaire à un fichier binaire.

* * * * * *

Il nous reste maintenant à étudier le programme d'utilisation du fichier.

En voici un exemple très simple. On pourra s'inspirer du dernier programme d'utilisation développé plus haut pour améliorer la présentation des caractères graphiques.

```
10 CLEAR ,48126,1

20 CLS:LOCATE 7,12:
   PRINT "NOM DU FICHIER DE STOCKAGE"

30 LOCATE 15,16:INPUT F$

40 LOADM F$


100 M0=49151:NG=PEEK(M0)

110 FOR I=0 TO NG

120 CLS:COLOR 1:LOCATE 19,3:
    PRINT I;:COLOR 4:LOCATE 0,7

130 FOR J=1 TO 8

140 Z(J)=PEEK(M0-8*I-J):PRINT Z(J)

150 NEXT J

160 DEFGR$(0)=Z(1),Z(2),Z(3),Z(4),
              Z(5),Z(6),Z(7),Z(8)

170 LOCATE 20,15:COLOR 1:PRINT GR$(0):COLOR 4

180 LOCATE 20,22:PRINT "TAPEZ ";:
    COLOR 0,3:PRINT "ENTREE":COLOR 4,6

190 R$=INPUT$(1):IF ASC(R$)<>13 GOTO 190

200 NEXT I:CLS
```

*** 10
Protège les mémoires, de l'adresse 48126 à l'adresse 49151, contre une utilisation par l'interprèteur BASIC et réserve une place pour un caractère graphique.

On peut s'étonner de ne voir réserver qu'une seule place pour les caractères graphiques. En réalité nous bâtirons le caractère GR$(0) à l'aide des huit données du caractère I, sur appel de ce caractère et nous afficherons le caractère GR$(0). Il est donc nécessaire de ne réserver qu'une place pour GR$(0).

*** 20 - 30
Chargement par l'opérateur, du nom du fichier de stockage des données dans la variable chaîne de caractères F$.

*** 40
Charge le fichier dont le nom est stocké dans F$.

On notera que l'instruction est LOADM et non LOAD comme pour les fichiers BASIC.

*** 100
Le contenu de la mémoire d'adresse M0 = 49151 est le nombre NG de caractères stockés.

*** 110
Lancement d'une boucle en I (de 0 à NG)

*** 120 - 150
Boucle en J permettant de lire les huit nombres caractéristiques du caractère I. On notera que la formule donnant l'adresse de la mémoire de stockage de Z(J) est la même que celle du programme de création.

*** 160
Définition du caractère GR$(0) avec les nombres Z(1), ...Z(8) du caractère I.

*** 170
Affichage en Rouge du caractère GR$(0).

*** 180 - 200
Instructions permettant à l'opérateur de commander l'affichage du caractère suivant.

Une fois le dernier caractère affiché l'écran est effacé.

Cette méthode de stockage de données n'est bien entendu valable que pour les variables dont la valeur est comprise entre 0 et 255.

Il est toutefois possible de stocker des chaînes de caractères à condition de représenter chaque caractère par son numéro de code ASCII et de séparer les chaînes par un caractère spécial, par exemple le caractère "ENTREE" de numéro de code 13. La relecture des chaînes en mémoire se fait alors par des instructions PEEK successives jusqu'à ce qu'une valeur 13 soit rencontrée. Chaque valeur lue est bien entendu transformée en caractère par la fonction CHR$. Voici deux courts programmes exemples de chargement et de relecture d'un fichier de chaînes de caractères.

```
10 CLEAR ,49000

20 DATA Lundi,Mardi,Mercredi,Jeudi,
       Vendredi;Samedi,Dimanche

30 M=49151

40 FOR I=1 TO 7

50 READ Z$

60 FOR J=1 TO LEN(Z$)

70 POKE M,ASC(MID$(Z$,J,1)):M=M-1

80 NEXT J

90 POKE M,13:M=M-1

100 NEXT I

110 POKE M,0

120 CLS:LOCATE 2,12:
    PRINT "Le fichier est stocké par la commande":
    LOCATE 5,14:PRINT "SAVEM ";CHR$(34);
    "JOURS";CHR$(34);",";M;",49151,0"

130 SAVEM "JOURS",M,49151,0
```

*** 10
Les mémoires de 49000 à 49151 sont protégées.

*** 20
Liste de chaînes de caractères que l'on s'apprète à stocker dans le fichier binaire.

*** 30
Initialisation à 49151 de la mémoire M de stockage des données. Les données seront placées dans les mémoires dans le sens décroissant des adresses.

*** 40
Initialisation d'une boucle en I (de 1 à 7) destinée à lire les sept chaînes de caractères placées dans les DATA sous l'étiquette 20.

*** 60 - 80
Boucle en J destinée à analyser caractère par caractère la chaîne Z$, précédemment lue dans les DATA.

J varie de 1 à la longueur de la chaîne de caractères Z$. La fonction MID$(Z$,J,1) isole le caractère de rang J dans Z$ -(M.Ref 130)-, et la mémoire M est chargée avec le numéro de code ASCII (fonction ASC -(M.Ref 111)-) de ce caractère par la fonction POKE -(M.Ref 86)-

Après chaque chargement l'adresse M de la mémoire de stockage est décrémentée de 1.

*** 90
Une fois chargés les numéros de code ASCII de tous les caractères d'une chaîne, la mémoire d'adresse suivante est chargée avec le code de séparation de chaînes, c'est-à-dire le code 13 (touche ENTREE). L'adresse M de la mémoire de stockage est bien entendu décrémentée pour le stockage du premier caractère de la chaîne suivante.

*** 100
Fermeture de la boucle en I.

*** 110
Chargement, dans la dernière mémoire d'adresse M, du code 0 de fin d'enregistrement (nous avons choisi 0 conventionnellement, sachant qu'aucun caractère n'a zéro comme code ASCII).

*** 120
Message permettant à l'opérateur de connaître les bornes de stockage du fichier "JOURS".

On notera la façon dont sont représentés les guillemets autour du mot JOUR dans la seconde chaîne affichée. On ne peut en effet mettre des guillemets dans une chaîne autrement qu'en les nomant par leur numéro de code ASCII (34).

Une fois le programme exécuté, on constate que l'adresse de chargement du code 0 de fin d'enregistrement est 40099. On a donc été large en réservant en 10 les mémoires à partir de 49000. Il aurait suffit de compter les données sur nos doigts pour prévoir ce résultat !

*** 130
Commande de stockage du fichier binaire JOURS entre M (= 49090) et 49151.

* * * * * *

Il nous faut maintenant bâtir le programme de relecture. En voici un exemple qui n'a pour but que de montrer la technique de récupération des données dans les mémoires par des instructions PEEK.

```
10 CLS:CLEAR ,49000

20 LOADM "JOURS"

30 M=49151

40 Z$=""

50 U=PEEK(M):M=M-1:
   ON 1-(U=13)-2*(R=0) GOTO 60,100,200

60 Z$=Z$+CHR$(U)

70 GOTO 50

100 PRINT Z$:GOTO 40

200 END
```

*** 10
Efface l'écran et réserve les places mémoire entre 49000 et 49151.

*** 20
Chargement du fichier binaire JOURS entre 49099 et 49151. Ces adresses sont mises en mémoire sur la cassette au moment du chargement et n'ont pas à être répétées à la lecture par LOADM.

On notera que le nom de fichier peut être omis, auquel cas le premier fichier binaire rencontré sera chargé. Il faut dans ce cas positionner correctement la bande magnétique de la cassette.

*** 30
Initialisation de l'adresse M de relecture à 49151.

*** 40
Initialisation à "zéro" de la chaîne de caractères Z$ dans laquelle on va stocker les chaînes relues.

*** 50
a/ U est la valeur numérique contenue dans la mémoire M (c'est-à-dire un numéro de code ASCII ou le code 13 de séparation ou le code 0 de fin de fichier).

b/ L'adresse M est décrémentée de 1.

c/ Si la valeur U est égale à 13 (séparation de chaîne), branchement en 100. Si la valeur de U est égale à 0 (fin d'enregistrement), branchement en 200. Dans les autres cas (U est alors un numéro de code ASCII), branchement en 60.

*** 60
U étant un numéro de code ASCII, le caractère correspondant CHR$(U) -(M.Ref 113)- est concaténé derrière Z$.

*** 70
Branchement en 50 pour la lecture de la mémoire suivante (dans le sens des adresses décroissantes).

*** 100
U ayant la valeur 13, la variable chaîne de caractères Z$ contient toute la chaîne stockée.

La valeur de Z$ est alors affichée pour contrôle et le programme branché en 40 pour le chargement d'une autre chaîne dans Z$.

On s'est contenté de présenter les chaînes les unes derrière les autres sans les stocker dans une variable indicée Z$(I). Nous laissons au lecteur le soin de modifier le programme dans ce sens (attention au dimensionnement du tableau Z$(I) par DIM dans le cas où le nombre de chaînes stockées est supérieur à 10).

On remarquera toutefois qu'il est dommage de disposer d'un premier stockage des chaînes entre 49099 et 49151 et de créer un nouveau stockage dans le tableau Z$(I). Il est préférable, dans la mesure où il n'est pas nécessaire de disposer de la chaîne numéro I, indépendemment des autres, de se contenter du premier stockage.

*** 200
U étant égal à 0, la fin de fichier a été détectée et le programme est stoppé.

* * * * * *

Le lecteur dispose maintenant des éléments lui permettant de voir les avantages et les limitations des deux types de stockage de données sur cassette.

Il prendra garde à placer ces fichiers sur la cassette juste après le programme d'utilisation de façon à y accéder rapidement et sûrement après chargement du programme et son exécution.

* * * * * *

On notera enfin qu'un fichier de données (caractérisé par le suffixe .DAT), ne peut être chargé directement par LOAD et donc ne peut être recopié sur une autre cassette.

La seule façon de charger le fichier données sur une autre cassette est d'exécuter à nouveau le programme de création du fichier.

Cette remarque suggère une méthode pour protéger les programmes contre des recopies pirates !

Une partie des données nécessaires au fonctionnement du programme est stockée dans un fichier données dans un ordre arbitraire, connu du seul auteur. La relecture du fichier ne peut se faire dans le programme que dans l'ordre d'écriture. Si le programme est sauvé sur la cassette par la commande SAVE "nom du fichier",P -(M.Ref 167)-, c'est-à-dire sous forme protégée, on sait que le programme est exécutable mais non listable. Le "pirate" éventuel ne peut alors connaître la structure du fichier et il ne peut donc en reconstituer le programme de création nécessaire pour le recopier sur une autre cassette.

On prendra garde toutefois à ne pas utiliser d'instructions PEEK et POKE dans le programme, car ces instructions ne peuvent figurer dans un programme protégé.

11

Calendrier

Nous allons nous donner les moyens d'établir un programme affichant à l'écran le calendrier de l'un des mois d'une année quelconque comprise entre 1901 et 2099. Il est possible de généraliser à une année hors de ce domaine, mais les calculs sont un peu plus compliqués et en pratique la période choisie est suffisante pour les besoins courants.

Pourquoi cette période ? Simplement parce que toutes les années divisibles par quatre sont bisextiles, même 2000 alors que 1900 et 2100 ne le sont pas.

Il suffit pour établir notre calendrier, de compter le nombre N de jours écoulés entre le premier du mois demandé et le premier Janvier 1901. Si le reste de la division de N par 7 est nul, c'est que le premier du mois tombe sur le même jour de la semaine que le premier Janvier 1901. Sinon le reste de la division par sept donne le décalage.

Mais direz-vous, il nous faut connaître quel jour tombait le premier Janvier 1901 !

Inutile en réalité, il suffit de recaler la formule en ajoutant à N une constante, comprise entre 0 et 6, de telle sorte que l'on obtienne un résultat correct pour un jour connu, par exemple Dimanche pour le premier Mai 1983.

Moyennant quoi il est possible de savoir que le premier Janvier 1901 était un Mardi !

Le nombre N de jours compris entre le premier Janvier 1901 et le premier du mois M (M variant de 1 à 12) est donné par :

$$N = INT ((A - 1901) * 365.25 + NM(M) + 1)$$

où NM(M) est le nombre de jours cumulés entre le premier Janvier et le dernier jour du mois précédent le mois M (en rajoutant .25 jours à partir de Mars et en comptant Février pour 28 jours, ce qui permet de tenir compte des années bisextiles). Les valeurs de NM(N) sont facilement calculables une fois pour toutes et figureront dans une table de DATA en début de programme.

La fonction INT(X) -(M.Ref 126)- calcule, rappelons-le, la partie entière de X.

La partie technique étant claire, il nous reste à présenter les résultats de façon harmonieuse.

Nous tracerons un tableau rectangulaire comportant sept lignes de trois lignes texte de hauteur, ce qui permet de placer les quantièmes du mois au centre des lignes tableau.

La première colonne du tableau contiendra les sept jours de la semaine en commençant par Dimanche.

Un mois de 31 jours peut s'étendre sur six semaines. Par exemple Octobre 1983 commence un Samedi et se termine un Lundi. Notre tableau comportera donc six colonnes de cinq caractères de large.

Nous tracerons en outre les verticales de séparation entre les colonnes du tableau à des abscisses non multiples de huit de façon à pouvoir les tracer sur l'imprimante thermique (les points d'abscisses multiples de huit ne sont pas reproduits sur l'imprimante thermique lorsque l'on recopie l'écran par la commande SCREENPRINT).

Nous déterminerons donc le jour de la semaine du premier du mois et placerons un 1 dans la première colonne à la ligne correspondante.

Nous remplirons ensuite les cases suivantes, de haut en bas, avec retour en haut de la colonne suivante, par les chiffres de 2 à 31 (ou à 28, 29, 30 suivant les mois et les années).

Ces principes étant admis voyons la réalisation pratique et les améliorations que l'on peut apporter au fur et à mesure de la mise au point.

Le programme comporte tout d'abord un bloc de chargement de données :

```
10 CLS:SCREEN 4,6,6

20 EF$="          ":SOS$="SORE"

30 DIM NM(12),M$(12)

40 DATA 0,31,59.25,90.25,120.25,151.25,181.25,
        212.25,243.25,273.25,304.25,334.25

50 FOR I=1 TO 12:READ NM(I):NEXT

60 DATA JANVIER,FEVRIER,MARS,AVRIL,MAI,JUIN,
        JUILLET,AOUT,SEPTEMBRE,OCTOBRE,
        NOVEMBRE,DECEMBRE

70 FOR I=1 TO 12:READ M$(I):NEXT

80 DATA Dimanche,Lundi,Mardi,Mercredi,
        Jeudi,Vendredi,Samedi

90 FOR I=1 TO 7:READ J$(I):NEXT

100 SC=0:R=0
```

10
a/ Efface l'écran

b/ Impose la couleur Bleu (4) pour les caractères et la couleur Cyan (6) pour le fond et l'entourage.

*** 20
a/ La variable EF$, composée de dix espaces servira de chaîne d'effacement en cas d'erreur de frappe.

b/ La variable SOS$, composée d'un SOL et d'un RE sera utilisée comme signal d'alarme ("PIN-PON"), avec l'instruction PLAY -(M.Ref 201)-, en cas d'erreur de frappe.

*** 30
Dimensionnement à 12 valeurs de la variable NM(I), nombre de jours cumulés dans les mois précédents le mois I et de la variable chaîne de caractère M$(I), nom du mois I.

*** 40 - 50
Chargement des 12 valeurs de la variable NM(I).

*** 60 - 70
Chargement des noms de mois dans les variables M$(I).

*** 80 - 90
Chargement des noms de jours de la semaine dans la variable J$(I).

*** 100
Initialisation à zéro des deux indicateurs SC et R.

SC sera posé égal à -1 lorsque la recopie du calendrier sur l'imprimante thermique aura été demandée (SC pour SCreenprint).

R sera posé égal à -1 lorsque l'opérateur demandera l'impression des douze mois de l'année (R pour Répétition).

* * * * * *

Ce bloc de données étant constitué, passons au recueil de l'année et du mois entrés par l'opérateur et au calcul du jour de semaine du premier du mois.

```
500 CLS:LOCATE 15,2:PRINT "CALENDRIER"

510 LOCATE 10,6:PRINT "Annee :":
    LOCATE 10,8:PRINT "Mois  :"

520 LOCATE 0,20:PRINT
    "Pour imprimer une annee complete tapez":
    LOCATE 0,22:
    PRINT "un numero de mois egal a 13"
```

```
530 K=1:LOCATE 18,6:PRINT EF$:LOCATE 18,6:
    LINEINPUT Z$:GOSUB 5000:
    IF E THEN A=Z:A$=Z$ ELSE PLAY SOS$:GOTO 530

540 IF A<1901 OR A>2099 THEN PLAY SOS$:GOTO 530

550 K=2:LOCATE 18,8:PRINT EF$:LOCATE 18,8:
    LINEINPUT Z$:GOSUB 5000:
    IF E THEN M=Z ELSE PLAY SOS$:GOTO 550

560 IF M>13 OR M=0 THEN PLAY SOS$:GOTO 550

570 IF M=13 THEN R=-1:SC=-1:M=1

580 GOSUB 4000:NJ=NN:M=M-1:IF M=0 THEN NJP=31:M=1
    ELSE GOSUB 4000:NJP=NN:M=M+1


600 N=INT((A-1901)*365.25+NM(M)+1)+1

610 IF N>32767 THEN N=N-32767:GOTO 610

620 J=N MOD 7


4000 NN=31+(M=4)+(M=6)+(M=9)+(M=11)+
     (M=2)*(3+(A MOD 4=0)):RETURN


5000 L=LEN(Z$):E=-1+(K=1)*(L<>4)+(K=2)*(L>2):
     IF NOT E THEN RETURN


5010 FOR I=1 TO L:U=ASC(MID$(Z$,I,1)):
     IF U<48 OR U>57 THEN E=0:RETURN ELSE NEXT


5020 Z=VAL(Z$):RETURN
```

*** 500
Affichage de "CALENDRIER" au centre de la ligne 2 écran.

*** 510
Affichage de "Année :" et de "Mois :" en lignes 6 et 8

*** 520
Affichage en lignes 20 et 22 du message "Pour imprimer une année complète tapez un numéro de mois égal à 13"

Ce "raffinement" a été ajouté, par l'auteur, une fois le programme mis au point. La commande ainsi prévue permet d'éditer le calendrier complet d'une année sans avoir à répéter l'année et la succession des mois. Ce n'est qu'après utilisation du programme que le besoin d'une telle possibilité est apparu.

*** 530
Le numéro de l'année est entré par l'instruction LINEINPUT et chargé dans la variable chaîne de caractères Z$.

Le sous-programme 5000 est destiné à tester la validité de la chaîne ainsi entrée. Il vérifie que la chaîne comporte 4 caractères qui ne sont que des chiffres à l'exclusion de tout autre caractère.

Comme ce sous-programme 5000 est également utilisé pour tester en 550 la validité de la chaîne entrée pour le mois (longueur de chaîne 1 ou 2), il est nécessaire de préciser avant l'appel par GOSUB 5000 que l'on teste l'année et non le mois. Le paramètre K est posé égal à 1 à cet effet. Nous en verrons l'utilisation dans le sous-programme 5000.

Si la chaîne Z$ est reconnue acceptable, le sous-programme 5000 pose le paramètre E égal à -1, une erreur conduisant à E = 0.

Dans ces conditions si E est égal à -1 (Vrai), la condition IF E est vérifiée; la variable A est alors posée égale à Z (valeur numérique de Z$ calculée dans le sous-programme 5000) et la variable A$ est posée égale à Z$.

Dans le cas contraire (erreur) l'instruction ELSE provoque le signal sonore d'erreur (PLAY SOS$) et renvoie au début de l'instruction 530. C'est dans ce cas que la deuxième et la troisième instructions de 530 sont utiles, puisqu'elles effacent la chaîne refusée.

*** 540

Si la valeur A (année) n'est pas comprise dans les limites de validité (1901 et 2099) le signal d'erreur est "joué" par PLAY SDS$ et le programme rebranché en 530 pour effacement de la valeur affichée et nouvelle lecture de donnée.

*** 550

Ces instructions sont analogues aux instructions de 530. Elles sont destinées à recueillir le numéro du mois. On signale au sous-programme 5000 que c'est un numéro de mois qui est à tester en posant K = 2.

*** 560

Instruction analogue à 540 pour vérifier que le numéro de mois est compris entre 1 et 13.

N'oublions pas que le numéro 13 est la commande d'impression du calendrier pour les douze mois de l'année.

*** 570

Si M est égal à 13 le paramètre R (Répétition) est posé égal à -1. C'est cette valeur qui provoquera la répétition de l'impression pour les douze mois de l'année.

En outre le paramètre SC (SCreenprint) est posé égal à -1. C'est cette valeur qui provoquera le branchement sur l'instruction d'impression.

Enfin le numéro de mois est remis à 1 pour commencer le calendrier au mois de Janvier.

*** 580

a/ Branchement au sous-programme 4000 qui calcule le nombre de jours NN du mois M.

b/ Le nombre de jours du mois M est stocké dans la variable NJ.

c/ Le numéro du mois est décrémenté de 1 pour calculer le nombre de jours du mois précédent.

d/ Si M ainsi décrémenté, est nul, c'est que l'opérateur a demandé le calendrier de Janvier. Le nombre de jours du mois précédent NJP est donc 31. Le numéro du mois demandé est alors rétabli à sa valeur initiale 1.

Si M n'est pas nul alors NJP est calculé par le sous-programme 4000 (avec M égal au numéro du mois précédent). Enfin le numéro du mois est rétabli à sa valeur initiale en incrémentant M de 1.

Nous verrons plus loin pourquoi il est nécessaire de calculer le nombre de jours NJP du mois précédent.

*** 600

N=INT((A-1901)*365.25+NM(M)+1)+Q est le nombre de jours écoulés entre le premier du mois demandé et le premier Janvier 1901 à une constante additive Q près.

Nous choisirons la constante Q de telle sorte que le reste de la division de N par 7 pour le Premier Mai 1983 (qui est un Dimanche) soit égal à zéro. Nous conviendrons en effet par convention que Dimanche est le jour 0 de la semaine. On constate que la valeur de Q est 1, d'où l'instruction 600.

*** 610 - 620

La fonction Modulo (MOD) -(M.Ref 19)- n'est applicable qu'aux nombres inférieurs à 32767. Toute tentative d'utilisation de MOD avec un nombre supérieur à 32767 se traduit par une erreur OV (OVerflow, dépassement de capacité).

Il se trouve que 32767 est un multiple de 7 (32767=4681x7). Avant d'utiliser la fonction MOD sur N (qui atteint la valeur 72655 pour Décembre 2099), nous diminuerons donc N de 32767 autant de fois que nécessaire pour obtenir une valeur inférieure à 32767. C'est le but de l'instruction 610.

L'instruction 620 fournit alors le reste J de la division de N par 7.

J est ainsi le numéro du jour de la semaine (0 pour Dimanche, 6 pour Samedi) du premier jour du mois.

*** 4000

Le sous-programme 4000 calcule le nombre de jours du mois M (M = 1 pour Janvier, M = 12 pour Décembre).

Les mois d'Avril (4), Juin (6), Septembre (9) et Novembre (11) n'ont que 30 jours et Février n'a que 28 jours ou 29 les années bissextiles (dans ce cas A est divisible par 4 et A MOD 4 est nul).

La formule donnant NN soustrait 1 de 31 lorsque les variables logiques (M = 4),..., (M=11) sont Vrai (-1), c'est-à-dire pour Avril, Juin, Septembre et Novembre.

Elle soustrait 3 de 31 pour le mois de Février, la variable logique (M=2) étant alors Vrai et seulement 2 pour le mois de Février des années bissextiles puisque la variable logique (A mod 4=0) est Vrai et vaut alors -1.

Enfin NN vaut 31 pour tous les autres mois.

*** 5000

a/ L est la longueur de la chaîne de caractères captée en 530 (alors l'indicateur K est égal à 1 et Z$ est le numéro de l'année) ou en 550 (alors l'indicateur K est égal à 2 et Z$ est le numéro du mois).

b/ Le signal d'erreur E est égal à zéro dans l'un des deux cas suivants:

- K est égal à 1 (la variable logique (K=1) vaut -1,Vrai) ET Z$ n'a pas quatre caractères (la variable logique (L<>4) vaut -1, Vrai).

  Dans ce cas (K=1)*(L<>4) vaut 1 et E est nul.

- K est égal à 2 (la variable logique (K=2) vaut -1,Vrai) ET Z$ a plus de deux caractères (la variable logique (L>2) vaut -1, Vrai).

  Dans ce cas (K=2)*(L>2) vaut 1 et E est nul.

Dans les autres cas E vaut -1.

c/ Si E est égal à zéro (il y a une erreur sur la longueur de Z$), alors E est faux et NOT E est vrai. La condition derrière le IF est remplie et RETURN provoque la sortie du sous-programme.

La valeur E = 0 sera alors exploitée en 530 ou en 550 comme signal d'erreur.

*** 5010
Boucle en I, variant de 1 à la longueur L de Z$, permettant d'explorer les caractères de Z$.

MID$(Z$,I,1) est le caractère de rang I de Z$ et U son numéro de code ASCII. Si ce numéro de code n'est pas compris entre 48 (code de 0) et 57 (code de 9) alors le signal d'erreur E est mis à zéro et la sortie du sous-programme provoquée par RETURN, sinon la boucle d'exploration se poursuit.

*** 5020
Aucune erreur n'a été détectée (E vaut -1) et Z est la valeur numérique de Z$ (on est sûr que Z$ représente un nombre entier puisque composé uniquement de chiffres). Il y a alors sortie du sous-programme avec E = -1.

Les numéros d'année A et de mois M ont ainsi été entrés et l'on a vérifié qu'ils étaient dans le domaine acceptable. Le numéro J du jour de la semaine du premier du mois, les nombres NJ et NJP de jours dans le mois et dans le mois précédents ont été alors calculés. Il nous reste maintenant à dessiner le calendrier sur l'écran.

Bien entendu les instructions 10 à 620 constituent un 8loc exécutable.Il faut donc le tester avant de passer à la suite du programme.Il suffit pour cela d'écrire des instructions provisoires en 700, pour afficher les valeurs de A, M, N, NJ et NJP.

La partie dessin du calendrier se présente de la façon suivante :

```
1000 CLS:LOCATE 20-(LEN(M$(M))+5)/2,1,0:
     PRINT M$(M)+" "+A$

1010 BOXF(0,24)-(79,47),-2

1020 BOXF(0,48)-(79,192),-4
```

```
1030 LINE(0,24)-(0,192),0:
     LINE(1,24)-(1,192),0

1040 FOR X=79 TO 319 STEP 40:
     LINE(X,24)-(X,192),0:NEXT

1050 FOR Y=24 TO 192 STEP 24:
     LINE(0,Y)-(319,Y),0:NEXT

1060 COLOR 7,1:LOCATE 1,4:PRINT J$(1)

1070 COLOR 0,3:FOR I=2 TO 7:
     LOCATE 1,3*I+1,0:PRINT J$(I):NEXT

1080 COLOR 4,6

1090 IF R THEN LINE(0,0)-(319,0),0



2000 CC=12:LL=4+3*J

2010 FOR IM=1 TO NJ

2020 LOCATE CC,LL:PRINT USING "##";IM

2030 LL=LL+3:IF LL=25 THEN LL=4:CC=CC+5

2040 NEXT

2050 IF R GOTO 2500



2100 N=1:COLOR 5

2110 IF LL=25 THEN LL=4:CC=CC+5:
     IF CC=42 GOTO 2200

2120 LOCATE CC,LL:PRINT USING "##";N

2130 LL=LL+3:N=N+1:GOTO 2110
```

```
2200 COLOR 1:N=NJP:CC=12:LL=1+3*J

2210 IF LL=1 GOTO 2500

2220 LOCATE CC,LL:PRINT USING "##";N

2230 LL=LL-3:N=N-1:GOTO 2210


2500 COLOR 4:IF SC THEN SCREENPRINT:
     IF R GOTO 3500 ELSE GOTO 3010


3000 LOCATE 33,24:COLOR 0,3:PRINT "ENTREE";:
     COLOR 4,6:R$=INPUT$(1):
     IF ASC(R$) () 13 GOTO 3000

3010 CLS:LOCATE 5,12:
     PRINT "Imprimer le calendrier tapez";:
     COLOR 0,1:PRINT "I":COLOR 4,6

3020  LOCATE 5,10:PRINT "Pour":LOCATE 5,14:
      PRINT "Un autre calendrier    tapez";:
      COLOR 0,1:PRINT "A":COLOR 4,6

3030 LOCATE 5,16:
     PRINT "Cloturer                tapez";:
     COLOR 0,1:PRINT "C":COLOR 4,6

3040 R$=INPUT$(1)

3050 ON 1+(R$="I")*(M(13)-2*(R$="A")-3*(R$="C")
     GOTO 3040,3100,100,3200


3100 SC=-1:GOTO 1000


3200 CLS:END


3500 M=M+1:IF M=13 THEN CLS:
     GOTO 3020 ELSE GOTO 580
```

*** 1000

Affiche, après avoir effacé l'écran, le nom du mois et le numéro de l'année, au milieu de la ligne écran 1.

Le curseur est effacé par le zéro en troisième paramètre de l'instruction LOCATE.

Le calcul du numéro de colonne dans l'instruction LOCATE tient compte de la longueur du nom de mois, par LEN(M$(M)) et de la longueur du numéro de l'année (quatre plus un espace de séparation).

*** 1010

Place une plage rectangulaire, en couleur Rouge fond (-2), sur les dix premières colonnes, de la ligne 3 à la ligne 5 écran.

Il est nécessaire de colorer cette plage en couleur fond puisque nous allons écrire dessus la chaîne de caractères "Dimanche".

*** 1020

Place une plage rectangulaire, en couleur Jaune fond (-4), sur les dix premières colonnes, de la ligne 6 à la ligne 23 écran.

Il est nécessaire de colorer cette plage en couleur fond puisque nous allons écrire dessus les chaînes de caractères "Lundi" à "Samedi".

*** 1030

Ces deux instructions tracent deux verticales en Noir caractère (0) d'abscisses 0 et 1. La seconde sera la seule recopiée par l'instruction SCREENPRINT (les points d'abscisses multiples de huit ne sont pas recopiés).

Il est nécessaire de placer la première verticale pour éviter d'avoir une ligne verticale colorée en Rouge puis en Jaune sur le bord gauche de l'écran.

*** 1040

Tracé en Noir des sept verticales de séparation des colonnes du tableau.

Les abscisses sont des multiples de huit moins 1.

*** 1050
Tracé en Noir des huit horizontales de séparation des lignes tableau.

*** 1060
Affichage dans la première colonne, première ligne du tableau de la chaîne de caractères J$(1), c'est-à-dire "Dimanche" en Blanc (7) sur fond Rouge (1).

*** 1070
Affichage des jours de la semaine de "Lundi" à "Samedi" dans la première colonne du tableau, de la ligne 2 à la ligne 7 tableau.

La couleur des caractères est Noir (0) sur fond Jaune (3).

*** 1080
La couleur des caractères est ramenée au Bleu (4) sur fond Cyan (6).

*** 1090
Si R est égal à -1 (ce qui signifie que la répétition des douze mois du calendrier a été demandée), alors la condition R derrière IF est Vrai et une horizontale est tracée tout en haut de l'écran.

Ici on trouve un raffinnement qui n'a été rajouté qu'une fois essayé le programme.

Lorsque l'on demande le calendrier des douzes mois de l'année, les tableaux correspondants à chaque mois sont reproduits automatiquement sur l'imprimante (l'indicateur SC est mis égal à -1 en même temps que R en 570 si l'opérateur a affiché la commande 13 pour le numéro de mois).

Il est commode de séparer les différents tableaux par des horizontales, ce qui permet de plier régulièrement, en accordéon, la bande de papier occupée par les douze tableaux.

La ligne horizontale tracée en haut du tableau joue alors ce rôle, une fois reproduite sur la bande de l'imprimante.

*** 2000

Le premier jour du mois doit être placé en première colonne tableau (colonne écran CC = 12), sur la ligne tableau du jour J de la semaine (J = 0 pour le Dimanche, J = 6 pour le Samedi). La chaîne de caractères Dimanche est placée en ligne 4 écran et les jours de la semaine sont placés toutes les trois lignes, d'où la formule donnant la ligne écran LL où l'on va afficher le premier jour du mois IM = 1.

*** 2010

Ouverture d'une boucle en IM de 1 à NJ, nombre de jours du mois demandé.

*** 2020 - 2030

Affichage de IM, en colonne écran CC, ligne écran LL.

L'instruction PRINT USING avec une image comportant deux caractères dièse permet d'afficher les quantièmes du mois en les justifiant à droite (les chiffres des unités sont sur la même colonne écran, que les nombres aient un ou deux chiffres).

Le numéro de ligne écran LL est ensuite incrémenté de trois, pour afficher le nombre IM suivant sur la ligne tableau du dessous. Toutefois si LL atteint 25, c'est-à-dire si le dernier affichage a été effectué sur la dernière ligne tableau (en ligne 22 écran), alors LL est ramené à 4 (ligne écran correspondant à la ligne tableau de Dimanche) et CC incrémenté de 5, de façon à inscrire les quantièmes suivants sur la prochaine colonne tableau (les colonnes tableau ont 5 caractères de large).

*** 2040

Fermeture de la boucle en IM.

*** 2050

Les instructions de 2100 à 2230 sont destinées à compléter le tableau avec les quantièmes des mois précédent et suivant. Il est en effet commode de disposer de ces renseignements complémentaires lorsque l'on ne demande que le calendrier d'un seul mois. Par contre ce complément est inutile lorsque l'on édite le calendrier complet de l'année (l'indicateur R est alors égal à -1).

L'instruction 2050 courcircuite ainsi ces instructions lorsque R est Vrai en branchant le programme en 2500.

*** 2100 - 2130
Après affichage du dernier quantième du mois NJ, le numéro de ligne écran LL a été incrémenté de 3 par l'instruction 2030 (ou ramené à 4 si l'affichage de NJ avait été fait sur la dernière ligne tableau). CC et LL sont alors les coordonnées d'affichage du premier du mois suivant.

Les instructions 2100 à 2130 affichent alors en Magenta (5), les quantièmes du mois suivant à partir de cette position.

L'instruction 2120 affiche le quantième N, initialisé à 1 en 2100 et incrémenté de 1 en 2130.

Le numéro de ligne écran LL est incrémenté de 3 en 2130 et le programme branché en 2110 où LL est ramené à 4 s'il atteint 25 avec incrémentation de 5 pour CC. Un test sur CC (CC = 42) permet de détecter le remplissage de la totalité de la dernière colonne tableau et commande le branchement en 2200.

*** 2200 - 2230
Ces instructions permettent le remplissage du début du tableau avec les quantièmes du mois précédent. Voilà pourquoi nous avons calculé le nombre de jours NJP du mois précédent.

La couleur est fixée au Rouge (1), le quantième N au nombre de jours NJP du mois précédent.

NJP doit être affiché dans la case tableau au-dessus de la case contenant le premier du mois, c'est-à-dire en colonne écran CC=12 et ligne écran LL = 1 + 3 * J, puisque le premier jour du mois est affiché en 4 + 3 * J.

Le quantième N est affiché en CC,LL par l'instruction 2220, puis LL est décrementé de 3 et N de 1 en 2230, avec branchement en 2210 où LL est testé. Si LL est égal à 1, c'est que la ligne tableau Dimanche a été remplie et le programme est branché en 2500.

On notera que le test sur LL est placé en 2210 avant l'affichage de N en 2220, pour éviter un affichage intempestif en ligne 1 dans le cas où le premier du mois tombe un Dimanche (le tableau est alors entièrement rempli à gauche par les quantièmes du mois demandé).

*** 2500

a/ La couleur caractère est fixée au Bleu (4).

b/ Si SC est Vrai (-1), impression demandée, ce qui est automatique si le calendrier de l'année complète a été commandé par le numéro de mois posé égal à 13 par l'opérateur ou qui résulte d'un choix ultérieur de l'opérateur (voir instructions 3000 - 3050), alors l'instruction SCREENPRINT recopie l'écran sur l'imprimante thermique. Rappelons que cette instruction est inutilisable avec l'imprimante à aiguille.

Si la recopie d'écran a été commandée et en conséquence effectuée, un test est fait sur R.

Si R est Vrai (-1), c'est-à-dire si l'impression du calendrier complet de l'année a été demandé, alors le programme est branché en 3500 où le numéro du mois est incrémenté de 1 pour calcul et impression du mois suivant (voir 3500).

Sinon (R est Faux, 0) le programme est branché en 3010 pour affichage d'un menu.

On notera que ce test sur R n'a lieu que si le test sur SC est positif, donc s'il y a recopie d'écran. Si la recopie d'écran n'a pas été demandée, alors le programme continue en séquence à l'instruction 3000.

*** 3000

Affichage du message "ENTREE" en bas et à droite de l'écran.

On notera le point-virgule en fin de l'instruction PRINT; son oubli provoque un saut de ligne et comme le message est écrit sur la ligne 24 le saut de ligne se traduit par une remontée intempestive de l'écran !

La couleur caractère qui avait été fixée au Noir (0) sur fond Jaune (3), pour bien simuler la touche "ENTREE" est ramenée au Bleu (4) sur fond Cyan (6).

Le programme est ensuite stoppé en attente de la frappe de la touche ENTREE et de cette seule touche afin de permettre à l'opérateur de consulter le calendrier.

*** 3010 - 3030
Affichage d'un menu, proposant à l'opérateur, l'impression (touche I), le calcul d'un autre calendrier (touche A) ou la clôture (touche C).

On notera que les touches à utiliser pour le choix sont affichées en Noir (0) sur fond Rouge (1), pour mieux ressortir sur l'écran.

Lorsque le mode impression du calendrier de toute l'année a été sélectionné, l'affichage du menu se fait à l'instruction 3500, quand l'incrémentation du numéro de mois M conduit à M = 13 (voir étiquette 3500). Comme le choix ne peut être qu'entre un autre calcul ou la clôture, il est inutile de proposer l'impression. Aussi ce retour se fait-il en 3020 et non en 3010. C'est pourquoi l'affichage du mot "Pour" en ligne 10 ne se fait qu'à l'instruction 3020 de façon à être présent même avec le menu réduit.

Bien entendu toutes ces "finesses" de programmation n'ont pas été prévues avant les essais d'utilisation. Ce n'est qu'à l'usage que l'on voit apparaître ces besoins de modifications et de raffinements du programme initial.

*** 3040
Arrêt du programme en attente de la réponse de l'opérateur au choix proposé dans le menu.

*** 3050
Si la réponse R$ est un A (autre calcul), le paramètre de branchement vaut 3 (la variable logique (R$="A") vaut -1) et le programme est branché sur la troisième étiquette soit 100. On notera qu'à cette étiquette 100 les deux indicateurs SC et R sont réinitialisés à zéro.

Si la réponse est un C (clôture), le paramètre de branchement vaut 4 (la variable logique (R$="C") vaut -1) et le programme est branché à la quatrième étiquette soit 3200.

Si la réponse est un I et si le numéro du mois choisi est inférieur à 13, alors les deux variables logiques (R$="I") et (M<13) sont égales à -1 et leur produit égal à 1. Le paramètre de branchement vaut 2 et le programme est branché à la deuxième étiquette soit 3100. On remarquera le signe + devant le produit pour tenir compte du signe positif lorsque les deux variables logiques sont égales à Vrai.

Dans le cas où M est égal à 13 (c'est ce qui se produit à la fin d'impression du calendrier d'une année complète) alors le produit (R$="I")*(M(13) est nul même si la touche I a été frappée. Ceci rend inopérante la touche I après impression du calendrier complet (le menu ne propose plus l'impression, comme on l'a vu dans les commentaires de 3010 - 3030; il faut donc rendre la touche I inopérante dans ce cas).

Enfin dans tous les cas d'utilisation d'une touche autre que A, C ou I (si l'on imprime le calendrier d'un seul mois), le paramètre de branchement vaut 1, toutes les variables logiques étant nulles et le programme est branché à l'étiquette 3040 pour une nouvelle lecture de R$.

*** 3100

La recopie de l'écran sur l'imprimante a été commandée par frappe de la touche I.

L'indicateur SC est posé égal à -1 (Vrai) et le programme branché en 1000 pour réaffichage du calendrier.

Le programme atteint alors l'étiquette 2500 où la condition IF SC étant vérifiée, l'instruction SCREENPRINT est exécutée.

*** 3200

La clôture a été demandée par frappe de la touche A.

L'écran est effacé et le programme est stoppé par l'instruction END.

*** 3500

Cette étiquette est atteinte lorsqu'à l'étiquette 2500 l'instruction SCREENPRINT a été exécutée (IF SC est vérifié) et que l'indicateur R de répétition est égal à -1 (Vrai). Nous sommes alors en mode d'impression du calendrier d'une année complète.

Le numéro du mois est incrémenté de 1.

Si le numéro de mois atteint la valeur 13 (tout le calendrier a été imprimé), l'écran est effacé et le programme branché en 3020 pour affichage du menu réduit. Il faut effacer l'écran avant le branchement car l'étiquette 3020 ne comporte pas d'instruction CLS (celle-ci est en 3010 pour l'affichage du menu complet).

Si M est inférieur à 13 alors (ELSE) le programme est branché en 580 pour le calcul et l'affichage du mois suivant.

* * * * * *

La vérification des branchements, des affichages corrects du calendrier dans tous les cas possibles et des divers modes de fonctionnement (mois isolés ou calendrier d'une année complète), nécessite, une fois tapé le programme tel qu'il a été conçu ab initio de rajouter quelques instructions complémentaires facilitant cette vérification. Nous allons en voir quelques exemples.

Il faut tout d'abord vérifier que l'affichage du calendrier est correct, dans tous les cas possibles :

- premier du mois un jour quelconque de la semaine du Dimanche au Samedi)
- nombre de jours du mois compris entre 28 et 31
- nombre de jours du mois précédent compris entre 28 et 31

Il n'est pas possible pratiquement de faire tourner directement le programme en choisissant au hasard les années et les mois de façon à couvrir tous ces cas.

Nous allons donc rajouter les instructions suivantes pour la mise au point (en n'oubliant pas de les supprimer pour le programme définitif).

```
95 A$="2000":GOTO 800

800 CLS:PRINT "M=":PRINT "J=":
    PRINT "NJ=":PRINT "NJP="

810 LOCATE 3,0:INPUT M:LOCATE 3,1:INPUT J
    LOCATE 3,2:INPUT NJ:LOCATE 3,3:INPUT NJP
```

```
820 SC=0:R=0

3005 GOTO 800
```

*** 95

Fixe le variable chaîne de caractères A$ à "2000" (pourquoi pes !). Cela permet de vérifier que l'effichage du mois et de l'année est correctement centré sur la ligne 1 de l'écran (étiquette 1000).

Le branchement en 800 permet de sauter les calculs de J, NJ et NJP.

*** 800 - 810

Les variables M (numéro du mois, pour l'affichege du nom de mois M$(M) en ligne 1, étiquette 1000), J (jour de la semaine du premier du mois evec J = 0 pour Oimanche et J = 6 pour Samedi), NJ (nombre de jours du mois) et NJP (nombre de jours du mois précédent) sont entrés eu clavier.

Comme ces instructions ne sont que provisoires, nous n'avons pas rajouté de tests de vraisemblance. Rien ne nous empèchera d'essayer un mois de Février de 32 jours précédé d'un mois de 25 jours !

Il nous reste quand même à ne pas essayer un jour de semaine supérieur à 6.

*** 820

Les indicateurs SC et R sont fixés définitivement à 0 (nous ne testons que l'affichage du calendrier sur un mois).

*** 3005

Le programme a été stoppé par l'instruction 3000. L'instruction provisoire 3005 branche en 800 pour un essai avec de nouvelles valeurs.

Ces instructions nous permettront de tester le programme de 1000 à 3000 en particulier les valeurs de CC et LL et leurs calculs.

Si bien entendu le lecteur programme ces instructions provisoires avec le programme tel que donné ici, il constatera que tout marche correctement. Nous ne saurons trop lui conseiller de reconcevoir le programme à titre d'exercice, sans relire le programme fourni. C'est alors qu'il constatera combien les instructions de mise au point sont utiles !

Il nous faut enfin vérifier le fonctionnement des indicateurs SC et R.

Si nous laissons l'instruction SCREENPRINT en 2500, nous dépenserons beaucoup de papier et de temps en mise au point.

Aussi est-il recommandé de modifier provisoirement l'instruction 2500 :

```
2500 COLOR 4:IF SC THEN CLS:PRINT "SCREENPRINT"
     :R$=INPUT$(1):IF R GOTO 3500 ELSE GOTO 3010
```

Ainsi l'instruction SCREENPRINT est remplacée par l'affichage du message "SCREENPRINT" à l'écran préalablement effacé et le message peut être lu puisque le programme est stoppé par l'instruction R$=INPUT$(1). Il est alors possible de tester les divers modes de fonctionnement et les options proposées par le menu.

A 1

Contrôle des couleurs

La programmation des couleurs est simple si l'on a bien compris un certain nombre de principes découlant de la façon dont sont representés les graphismes sur l'écran.

Donnons tout d'abord quelques renseignements sur la composition de l'écran et les possibilités de coloration de ses points. Nous préciserons ensuite ces notions à l'aide d'exemples.

L'écran est constitué de 200 lignes graphiques, chaque ligne ayant 320 points.

Nous ferons la distinction entre une "Ligne Graphique" constituée de 320 points et une "Ligne Texte" constituée de huit lignes graphiques superposées sur lesquelles s'inscrivent les caractères. L'ordonnée Y d'une ligne graphique est comprise entre 0 et 199 alors que la cote L d'une ligne texte est comprise entre 0 et 24. Nous utiliserons la notation L pour la cote de la ligne texte dans les les instructions en "mode caractère" et la notation Y pour l'ordonnée d'une ligne graphique dans les instructions en "mode graphique".

Mais les points d'une ligne ne sont pas indépendants. Ils sont regroupés en 40 segments de 8 points (8 x 40 = 320). Les points d'un même segment ne peuvent avoir que deux couleurs choisies, par le programmeur, parmi les huit possibles.

Chaque point d'un segment peut être affecté, par programme,à l'une ou l'autre des deux catégories de couleur que nous appelerons A et B.

Par la suite l'une des huit couleurs possibles peut être attribuée à chacune de ces deux catégories et ceci indépendamment pour chaque segment d'une même ligne ou des autres lignes.

Interessons nous pour l'instant à un segment donné, per exemple pour fixer les idées, au segment d'ordonnée Y = 100 et constitué des huit points d'abscisses X = 40 à X = 47 (les ebscisses X de début de segment sont multiples de 8).

Le programmeur peut décider (nous verrons plus loin comment) d'affecter tels points du segment à la catégorie A et tels points à la catégorie B.

Par exemple les points sont ainsi répartis :

A A B A A B B B

Si le programmeur affecte le Rouge (R) à la catégorie A et le Vert (V) à la catégorie B, on observera le segment :

R R V R R V V V

Si maintenant on change (par le progremme) la valeur de la couleur des points A du segment (Bleu par exemple à la place de Rouge), on observera le segment :

B B V B B V V V

Ainsi le changement de couleur attribuée à la catégorie A affecte tous les points A.

Cette remarque évidente nous permettra d'expliquer quelques phénomènes apparemment inattendus que l'on peut observer par exemple dans le cas suivant.

Ayant programmé les couleurs A et B d'un segment et ayant précisé quels points du segment sont A et quels points sont B, une instruction ultérieure indépendante vient chenger la couleur d'un seul des points du segment par exemple un point A. Dens ce cas tous les points A du segment chengent de couleur. Nous en donnerons des exemples (croisement de lignes ou traversée de cadres pleins, programmés par BDXF, par des lignes de couleurs différentes...)

Pour des raisons que nous préciserons plus loin, les deux familles A et B portent les noms de COULEUR OE GRAPHISME et de COULEUR DE FOND.

N'y ettachons, pour l'instant, aucun sens particulier car nous verons qu'ils n'ont de réelle significetion que pour l'affichage des caractères.

Contentons nous de penser qu'il est possible d'affecter chaque point d'un segment à une famille A (couleur de graphisme) ou à une famille B (couleur de fond) et qu' un chengement de valeur de l'une des deux couleurs change tous les points de la famille.

L'instruction -(M.Ref 193)-

SCREEN P,Q,R,J

fixe les valeurs des couleurs pour tous les segments de l'écran.

Les paramètres P, Q et R peuvent prendre les valeurs suivantes :

0	Noir	4	Bleu
1	Rouge	5	Magenta (Violet)
2	Vert	6	Cyan (Bleu clair)
3	Jaune	7	Blanc

- Le paramètre P fixe la couleur des caractères qui seront affichés sur l'écran par les instructions PRINT. Les points de l'écran appartenant à un caractère seront alors tous considérés comme appartenant à la famille A. C'est pourquoi la famille A porte le nom de "couleur de graphisme".

- Le paremètre Q fixe la couleur de tous les autres points de l'écran. Tous les points n'appartenant pas à un caractère feront partie de la famille B. La famille B porte ainsi le nom de "couleur de fond".

- Le paramètre R fixe la couleur du cadre entourant la partie utile de l'écran. Elle est en général fixée en début de programme et, sauf effets spéciaux, n'est pas modifiée en cours de programme.

- Enfin le paramètre J (valeur 0 ou 1, nous disons bien 0 ou 1, la valeur n'ayant pas de signification, seule la présence du paramètre en ayant une), intervertit les couleurs affectées aux familles A et B.

Ainsi l'instruction SCREEN fixe globalement l'affectation des points de l'écran aux familles A et B et les couleurs de chacune de ces familles.

Nous allons voir maintenant comment localement modifier ces affectations.

L'instruction -(M.Ref 189)-

COLOR PP,QQ,J

modifie les deux couleurs pour les caractères qui seront affichés par les instructions PRINT qui suivront.

La couleur PP sera affectée aux points des caractères et la couleur QQ aux autres points des carrés ou rectangles dans lesquels s'inscrivent les caractères (carrés de 8 x 8 points pour les caractères de taille normale,carrés 16 x 16 ou rectangles 8 x 16 et 16 x 8 pour les caractères de grande taille).

Ainsi une instruction COLOR modifie localement la couleur affectée aux points de la catégorie A (couleur de graphisme) et aux points de la catégorie B (couleur de fond) pour tous les segments appartenant aux carrés ou rectangles dans lesquels s'inscrivent les caractères affichés après cette instruction.

Mais il faut prendre garde à ce qu'une instruction SCREEN suivant diverses instructions COLOR viendra changer globalement la couleur des points de la catégorie A (donc les points appartenant aux caractères) et la couleur des points de la catégorie B (donc les points de fond).

Un exemple précisera ces notions.

Etudions le programme suivant :

```
10 CLS
20 SCREEN 4,6,6
30 LOCATE 0,0:PRINT "X"
40 COLOR 0,6
50 ATTRB 1,1:LOCATE 0,4:PRINT "Y"
60 COLOR 2,1
70 ATTRB 0,0:LOCATE 0,8:PRINT "Z"
```

*** 10
Efface l'écran

*** 20
Fixe la couleur Bleu (4) pour les caractères (couleur de graphisme), la couleur Cyan (Bleu clair) (6) (couleur de fond) pour l'écran,la couleur Cyan pour l'entourage.

*** 30
Affiche le caractère X dans le coin haut-gauche de l'écran. Le caractère est Bleu et les points du carré 8 x 8 entourant le caractère sont de couleur de fond, c'est-à-dire Cyan.

*** 40
Fixe la couleur des caractères (couleur de graphisme) au Noir (0) et la couleur des points des carrés entourant les caractères au Cyan (6).

Il aurait été possible de ne pas repréciser la couleur de fond puisqu'elle est identique à celle fixée par l'instruction SCREEN en 20. On aurait donc pu écrire l'instruction 40 sous la forme : 40 COLOR 0

*** 50
Affiche sur la cinquième ligne de texte, en colonne 0, le caractère Y en taille double.

La couleur des points du caractère est Noir. La couleur des points du carré 16 x 16 entourant le caractère est Cyan. Ces derniers points ne se distinguent pas des autres points d'écran.

*** 60
Fixe la couleur de graphisme à 2 (Vert) et la couleur de fond à 1 (Rouge).

*** 70
Affiche sur la huitième ligne de texte, en colonne 0, le caractère Z en taille simple.

Les points du caractère sont Vert (couleur de graphisme), les points du carré 8 x 8 dans lequel s'inscrit le caractère sont Rouge (couleur de fond). Ainsi les points du fond de caractère se distinguent des points du fond général écran.

Si une fois ce programme exécuté nous tapons directement l'instruction :

```
SCREEN 5,2,6
```

tous les caractères passeront en Magenta (5) qui est maintenant la couleur générale caractère et tous les points de l'écran n'appartenant pas à un caractère passeront au Vert (2) (couleur générale fond).

Les instructions présentées dans le court programme exemple sont des instructions dites en mode "caractère". Les points d'écran sont traités en "pavés" de 8 x 8 (ou 8 x 16, 16 x 8, 16 x 16 pour les caractères de taille double). Les instructions COLOR fixent ainsi, globalement, les catégories A (graphique) et B (fond) de tous les points d'un pavé. Nous allons étudier maintenant le mode graphique qui traite les points individuellement (avec quelques restrictions comme nous allons le voir)

Avant de passer au mode graphique rappelons qu'il est possible d'écrire de façon plus concise un ensemble COLOR,LOCATE,PRINT.

Ainsi les trois instructions figurant en 60 et 70 peuvent s'écrire -(M.Ref 187)-

```
PSET (0,8) "Z",2,1
```

Nous noterons que l'instruction PSET est une instruction mode caractère.

En conséquence les coordonnées figurant entre parenthèses sont des numéros de ligne texte (0 à 24) et de colonne (0 à 39). Le mode caractère est reconnu par la présence d'une chaîne figurant après la parenthèse.

Cette instruction ne permet d'afficher qu'un seul caractère (le premier de la chaîne si celle-ci contient plus d'un caractère).

Enfin les deux derniers paramètres sont les mêmes que ceux figurant dans un COLOR, le premier fixant la couleur de graphisme (catégorie A), le second fixant la couleur de fond (catégorie B). Il est même possible d'ajouter un dernier paramètre (indifféremment 0 ou 1), inversant les deux couleurs, comme dans un COLOR.

* * * * * *

Passons maintenant aux instructions graphiques permettant de traiter les points de l'écran "individuellement".

L'instruction de base est l'instruction -(M.Ref 187)-

PSET (X,Y),CC

qui permet de placer un point sur l'écran aux coordonnées X (X varie de 0 à 319) et Y (Y varie de 0 à 199) avec une couleur définie par le paramètre CC.

Il faut prendre garde à ce que la façon d'exprimer la valeur de CC permet de situer le point dans la catégorie des points A (ayant la couleur de graphisme) ou dans la catégorie des points B (ayant la couleur de fond) tout en précisant la valeur attribuée à cette couleur.

On a vu que la différence entre couleur de graphisme et couleur de fond est exprimée, dans une instruction SCREEN ou une instruction COLOR par la place du paramètre.

Dans ces deux instructions le premier paramètre attribue une valeur à la couleur de graphisme, le second attribue une valeur à la couleur de fond. Par ailleurs un point appartient d'office à la catégorie A (couleur de graphisme) si c'est un point d'un caractère et à la catégorie B (couleur de fond) dans le cas contraire.

Dans une instruction PSET qui n'agit que sur un point (défini par ses coordonnées X et Y), on précise deux choses à la fois :

- si l'on considère que le point appartient à la catégorie A ou à la catégorie B,

- quelle est la couleur attribuée à la catégorie retenue.

A cet effet le paramètre CC peut recevoir une valeur numérique comprise entre -8 et 7.

Si la valeur de CC est négative, alors le point est classé dans la catégorie B, c'est-à-dire dans les points ayant une couleur de fond.

Si la valeur est positive alors le point est classé dans la catégorie A, c'est-à-dire dans les points ayant une couleur de graphisme.

Le code des couleurs est le suivant (il n'est applicable que pour les instructions graphiques) :

Noir	0	ou	-1
Rouge	1	ou	-2
Vert	2	ou	-3
Jaune	3	ou	-4
Bleu	4	ou	-5
Magenta	5	ou	-6
Cyan	6	ou	-7
Blanc	7	ou	-8

Ainsi l'instruction

```
PSET (41,100),-8
```

considère le point de coordonnées X = 41, Y = 100 comme un point de catégorie B (couleur de fond) et lui attribue la couleur Blanc. Mais il faut prendre garde à ce que, ce faisant, tous les points du segment dans lequel se situe notre point (allant de X = 40 jusqu'à X = 47 de la ligne d'ordonnée Y = 100),classés, par des instructions précédentes, dans la catégorie B, se trouveront colorés en Blanc.

Si par exemple nous écrivons les deux instructions suivantes :

```
10 CLS:SCREEN 4,6,6

20 PSET (41,100),-8
```

nous obtiendrons bien un point Blanc en X = 41, Y = 100 mais il sera accompagné par tout le segment X = 40 à X = 47 de la ligne d'ordonnée Y = 100, puisque tous ces points ont été classés par l'instruction 10 dans la catégorie B (couleur de fond). La couleur de fond ayant été changée par l'instruction 20 (-8 = Blanc), tous les points du segment appartenant à la catégorie B passent à la nouvelle couleur.

Si donc nous voulons obtenir un point Blanc unique, il nous faut modifier l'instruction 20 par :

```
20 PSET (41,100),7
```

Dans ces conditions le point X = 41, Y = 100 se voit classé dans les points de catégorie A (couleur de graphisme) dont la couleur est passée au Blanc. Les autres points du segment étant dans la catégorie B (SCREEN en 10 les a automatiquement classés tels puisqu'il n'y a pas de caractère à cette place), restent de la couleur de fond imposée par SCREEN.

On obtient alors un point Blanc unique aux coordonnées 41,100.

Supposons maintenant que nous écrivions le programme suivant très voisin du précédent :

```
10 CLS:SCREEN 4,6,6

20 LOCATE 5,12:PRINT "XX"

30 PSET (41,100),1
```

L'instruction 10 fixe la couleur de graphisme (A) à 4 (Bleu) et la couleur de fond (B) à 6 (Cyan).

Ainsi l'instruction 20 affiche en ligne texte L = 12 (ordonnée Y = (12 + 1) x 6 = 104 pour la ligne graphique contenant les points bas des caractères) et en colonne C = 5 (abscisse 5 x 8 = 40 pour les points bord-gauche du premier caractère) la chaîne composée de deux X.

Les deux X apparaissent en Bleu sur fond Cyan.

L'instruction 30 place alors un point dans le carré 8 x 8 occupé par le premier caractère X. Elle attribue à ce point la catégorie A (couleur de graphisme) puisque le paramètre CC, égal à 1, est positif.

Le point apparait en Rouge (1) mais les points du premier X sur la ligne graphique Y = 100 passeront également au Rouge puisque ce sont des points de catégorie A (imposé par le SCREEN). Leur couleur doit donc changer pour celle imposée par l'instruction 30.

Par contre les points de la ligne graphique Y = 100 du second X ne sont pas modifiés puisqu'ils appartiennent à un segment indépendant du premier.

Cet effet néfaste de "bavure" d'une couleur sur l'autre ne peut être évitée qu'en attribuant au point défini par le PSET une couleur identique à celle déjà donnée pour les caractères, ce qui se fait très simplement en ne précisant pas la valeur du paramètre CC. Dans ce cas le point est considéré comme appartenant à la catégorie A (couleur de graphisme) et sa couleur est celle définie pour les points de sa catégorie par SCREEN ou par le dernier COLOR.

Une autre méthode consiste, si cela est possible, à placer le point en dehors du carré intéressé par le caractère !

On notera que la valeur prise par défaut dans un PSET est celle d'une couleur de graphisme définie par le dernier SCREEN ou le dernier COLOR (la couleur par défaut ne peut être une couleur de fond).

* * * * * * *

D'autres instructions permettent de définir des points. Ce sont -(M.Ref 187)-

```
BOX  (X1,Y1)-(X2,Y2),CC

BOXF (X1,Y1)-(X2,Y2),CC

LINE (X1,Y1)-(X2,Y2),CC
```

L'instruction BOX définit un cadre rectangulaire ayant pour coins les points X1,Y1 et X2,Y2.

L'instruction BOXF définit une plage colorée rectangulaire ayant pour coins les points X1,Y1 et X2,Y2.

L'instruction LINE définit une droite ayant pour extrémités les points X1,Y1 et X2,Y2.

Là encore le paramètre CC définit deux choses à la fois :

- si les points de la figure (cadre rectangulaire, plage rectangulaire, droite) appartiennent à la catégorie A (couleur de graphisme) ou à la catégorie B (couleur de fond),

- quelle est la couleur attribuée à la catégorie retenue.

Pour répondre à ce double objectif la convention est la même que pour l'instruction PSET.

On voit tout de suite les conséquences de ce type d'instruction. Il est nécessaire pour éviter les "bavures" d'un graphisme sur l'autre de les définir avec des catégories de couleurs différentes.

Le programme de démonstration suivant va préciser ces points.

```
10 CLS:SCREEN 4,6,6
20 FOR X=0 TO 312 STEP 8:PSET (X,40):NEXT
30 BOXF (100,50)-(132,82),-8
40 BOXF (100,100)-(132,132),7
50 PSET (0,0)"1",4,6
55 R$=INPUT$(1)
60 LINE (50,50)-(150,82),-1
70 LINE (50,60)-(150,92),0
80 PSET (0,0)"2",4,6
85 R$=INPUT$(1)
90 LINE (50,100)-(150,132),-1
100 LINE (50,110)-(150,142),0
110 PSET (0,0)"3",4,6
115 R$=INPUT$(1)
120 FOR X=80 TO 160 STEP 8
130 PSET (X,52),-1
140 PSET (X,55),0
150 PSET (X,102),-1
160 PSET (X,105),0
170 NEXT
180 PSET (0,0)"4",4,6
185 R$=INPUT$(1)
190 SCREEN 1,3,3
```

*** 10

Efface l'écran et fixe la couleur de graphisme à 4 (Bleu), la couleur de fond à 6 (Cyan) et la couleur de l'entourage à 6 (Cyan).

*** 20

Affiche 40 points aux abscisses multiples de 8 sur la ligne graphique Y = 40. Les points sont en couleur graphique Bleu (valeur prise par défaut).

*** 30 et 40

Ces deux instructions analogues définissent deux plages rectangulaires situées entre les lignes graphiques 50 et 82 pour la première, entre les lignes graphiques 100 et 132 pour la seconde.Les abscisses des coins sont les mêmes dans les deux cas, 100 pour le coin gauche, 132 pour le coin droit.Les deux plages sont Blanc (8 et -7), mais la première est en couleur B (couleur de fond) alors que la seconde est en couleur A (couleur de graphisme).

On constatera que le résultat n'est pas le même pour les deux plages. La première plage, en couleur de fond, débute à l'abscisse 96 et se termine à l'abscisse 135 (les points de 8 en 8 en ligne Y = 40 permettent de le vérifier). Par contre la deuxième plage, celle du bas, commence bien en X = 100 et se termine bien en X = 132.

La différence vient du choix du type de couleur.

Pour la plage supérieure les points d'abscisse 100 (à l'intérieur du segment X = 96 à X = 103) étant de couleur fond changent la couleur de tous les points fond du segment. Or les points de l'écran sont tous des points fond (puisqu'il n'y a pas de caractère). Les points de 96 à 99 passent donc au Blanc "entrainés" par les points 100 à 103. Il en est de même des points du segment X = 128 à X = 135 qui se colorent entièrement en Blanc puisque ce sont également des points fond.

Par contre la plage inférieure, tracée en couleur de graphisme ne déborde pas, les points 100 à 103 (couleur de graphisme) étant sans influence sur les points X = 96 à X = 99 (couleur de fond), de même que les points X = 128 à X = 132 (couleur de graphisme) sont sans influence sur les points X = 133 à X = 135

*** 50
Place le caractère "1" dans le coin haut-gauche de l'écran en couleur 4 (Bleu) sur fond 6 (Cyan)

*** 55
Attend qu'un caractère et un seul soit tapé au clavier et le place dans la variable R$.

Cette instruction permet d'arrêter le programme pour examiner le résultat et de reprendre le déroulement en tapant la touche ENTREE. Nous utiliserons des instructions analogues à 50 et 55 en 80,110 et 180 en changeant la valeur affichée dans le coin de l'écran pour se repérer dans le déroulement du programme.

*** 60 - 70
Ces deux instructions tracent deux droites parallèles qui traversent la plage Blanc supérieure.

Les deux droites sont tracées en Noir (-1 et 0), la première en couleur 8 (couleur de fond), la seconde en couleur A (couleur de graphisme).

On constate que la ligne supérieure (couleur de fond -1) "bave", chaque point de la ligne déclenchant la coloration Noir du segment auquel il appartient. Ceci est dû au fait que ligne et fonds, que ce soit l'écran ou la plage Blanc sont tous trois en couleur de fond (8).

Par contre la ligne inférieure est correcte puisqu'elle est en couleur (A) (couleur de graphisme) de nature différente des deux fonds traversés (couleur de fond 8).

*** 80 - 85
Voir 50 - 55.

*** 90 - 100
Instructions analogues à 60 - 70 définissant deux lignes Noir parallèles traversant la plage Blanc inférieure.

Là encore on constate que le graphisme est sans "bavure" dans les cas où droite et fonds n'appartiennent pas aux mêmes catégories de couleur, c'est-à-dire à l'intérieur de la plage Blanc pour la droite supérieure et à l'extérieur de la plage Blanc pour la droite inférieure.

*** 110 - 115
Voir 50 - 55

*** 120 - 170
Trace une suite de points espacés de 8 en 8 depuis l'abscisse X = 80 jusqu'à l'abscisse X = 160. Quatre lignes pointillées horizontales sont ainsi définies traversant les deux plages Blanc. La première et la troisième ligne sont en Noir couleur B (couleur de fond). La seconde et la quatrième sont en Noir couleur A (couleur de graphisme).

On constate que la première ligne est en fait continue, chaque point entraînant la coloration en Noir (B couleur de fond) du segment auquel il appartient. Par contre la deuxième ligne est correcte puisque les points sont en couleur A (couleur de graphisme) d'un type différent des deux fonds.

La troisième ligne est continue à l'extérieur de la plage Blanc inférieure (identité du type de couleur, les deux étant B couleur de fond) et correctement pointillée à l'intérieur de la plage Blanc (types différents de couleur, B pour la ligne et A pour la plage).

Ce résultat est évidemment inversé pour la dernière ligne.

*** 180 - 185
Voir 50 - 55

*** 190
Change les couleurs générales des points.

Tous les points ayant une couleur du type A (couleur de graphisme) passent en Rouge (1) et tous les points ayant une couleur de type B (couleur de fond) passent en Jaune (3).

Cela se traduit, en particulier, par la disparition de la plage supérieure (elle est maintenant en Jaune comme le reste de l'écran et n'est plus visible) ainsi que de tous les graphismes d'un type se détachant sur un fond du même type et que l'on ne distinguait que parce que les couleurs étaient différentes bien que de même type. Ainsi tous les graphismes qui "bavaient" ont disparu.

* * * * * *

Il est impossible de fixer des règles pratiques permettant d'éviter à coup sûr les "bavures" exceptée celle-ci :

NE PAS TRACER UN GRAPHISME AVEC UNE COULEUR OU MEME TYPE QUE CELUI OU FOND SUR LEQUEL IL APPARAIT.

On peut néanmoins se fixer une règle plus simple quitte à l'abandonner dans les cas un peu délicats pour revenir à la règle générale citée ci-dessus.

Ne définir les plages colorées qu'en couleur de fond (mais cela a l'inconvénient d'imposer des coins ayant des cotes multiples de 8 à gauche et multiples de 8 moins un à droite) et tracer les lignes en couleur de graphisme (mais attention aux bavures sur les caractères et les lignes voisines !)

Donnons un exemple de cas compliqué.

Nous désirons tracer une grille composée de neuf lignes horizontales Rouge depuis Y = 20 jusqu'à Y = 180, de neuf lignes verticales Bleu depuis X = 80 jusqu'à X = 240. Enfin nous placerons une plage Blanc au milieu de la grille avec son coin haut-gauche en X = 135, Y = 75 et son coin bas-droit en X = 185, Y = 125, le fond général écran étant Cyan.

Après quelques tâtonnements de mise au point on peut proposer le programme suivant :

```
10 CLS:SCREEN 4,6,6

20 BOXF (135,75)-(185,125),7

30 R$=INPUT$(1)

100 FOR Y = 20 TO 180 STEP 20

110 LINE (0,Y)-(319,Y),1

120 NEXT

130 R$=INPUT$(1)
```

```
200 FOR X = 40 TO 180 STEP 20
210 LINE (X,0)-(X,19),4
220 LINE (X,21)-(X,39)
230 LINE (X,41)-(X,59)
240 LINE (X,61)-(X,74)
250 LINE (X,75)-(X,125),-5
260 LINE (X,126)-(X,139),4
270 LINE (X,141)-(X,159)
280 LINE (X,161)-(X,179)
290 LINE (X,181)-(X,199)
300 NEXT
310 R$=INPUT$(1)

400 FOR I = 1 TO 6
410 X = 60+20*I-(I)3)*60
420 LINE (X,0)-(X,19),4
430 FOR Y = 21 TO 161 STEP 20
440 LINE (X,Y)-(X,Y+18)
450 NEXT Y
460 LINE (X,181)-(X,199)
470 NEXT I
480 R$=INPUT$(1)

500 SCREEN 4,6,6
```

*** 10
Efface l'écran. Les caractères seront Bleu et le fond Cyan avec un entourage Cyan.

*** 20
Affiche la plage Blanc. Il est nécessaire d'utiliser ici une couleur de graphisme puisque les abscisses des coins ne sont pas un multiple de 8, pour le coin gauche, ni un multiple de 8 moins un, pour le coin droit

*** 30
Le déroulement du programme s'arrête en 30 en attente de la frappe de la touche ENTREE. Cela permet de vérifier l'effet produit par les instructions précédentes.

*** 100 - 120
Boucle en Y affichant les lignes horizontales en Rouge. On a choisi une couleur de graphisme (Rouge 1), car la traversée de la plage Blanc ne pose pas de problème pour une ligne horizontale puisque les "bavures" ne se voient pas sur une ligne horizontale continue.

*** 130
Voir 30

*** 200 - 300
Boucle en X permettant d'afficher les verticales d'abscisses 140, 160 et 180.

On remarquera (210 - 230 et 270 - 280) que l'on ne trace les parties verticales, au-dessus et au-dessous de la plage, qu'entre les horizontales pour éviter qu'au croisement le Bleu ne vienne "baver" sur l'horizontale Rouge (verticales et horizontales sont toutes en couleur de graphisme).

Par contre dans la plage, une ligne verticale doit être tracée en couleur de fond puisque la plage est en couleur de graphisme. En conséquence plage et horizontales Rouge étant en couleur de graphisme, il n'y a pas de risque de "bavure" et l'on peut tracer directement les trois verticales d'un bord à l'autre de la plage, ce qui explique l'instruction 250.

*** 310
Voir 30

*** 400 - 470
Les verticales à l'extérieur de la plage (à gauche et à droite), doivent être en couleur de graphisme puisque se situant sur la partie Cyan de l'écran en couleur de fond. La croisée des horizontales doit se faire avec précaution comme en 210 - 240 et 260 - 290.

L'instruction 440 dans une boucle en Y permet de ne tracer les verticales qu'entre les horizontales sans les couper.

On notera que l'instruction 410 permet de calculer l'abscisse X de la ligne verticale numéro I, grâce à l'utilisation de la variable logique (I) 3). On obtient ainsi les six abscisses 80, 100, 120, 200, 220 et 240.

*** 480
Voir 50

*** 500
Ramène tous les points du type couleur de graphisme à la couleur Bleu et tous les points du type couleur de fond à la couleur Cyan.

Par suite la plage devient Bleu, ainsi que le réseau, extérieur à la plage, des horizontales et des verticales. Les horizontales intérieures sont noyées dans la couleur de graphisme et les verticales intérieures sont Cyan.

Une fois ce programme écrit et testé on peut se poser la question suivante :

Nous avons choisi de tracer les horizontales en couleur de graphisme Rouge (1). Pourquoi ?

Certainement par routine (il est recommandé de tracer une ligne avec une couleur du type différent du fond).

Or nous sommes dans le cas particulier de la ligne horizontale continue. Une telle ligne ne risque rien, quelle que soit la nature de sa couleur puisque les "bavures" ne se voient pas sur une horizontale continue.

Rebâtissons le programme avec les horizontales Rouge fond.

```
10 CLS:SCREEN 4,6,6
20 BOXF (135,75)-(185,125),7
30 R$=INPUT$(1)

100 FOR Y = 20 TO 180 STEP 20
110 LINE (0,Y)-(319,Y),-2
120 NEXT
130 R$=INPUT$(1)
200 FOR X = 40 TO 180 STEP 20
210 LINE (X,0)-(X,74),4
220 LINE (X,75)-(X,79),-5
230 LINE (X,81)-(X,99),-5
240 LINE (X,101)-(X,119),-5
250 LINE (X,121)-(X,125),-5
260 LINE (X,126)-(X,199),4
270 NEXT
280 R$=INPUT$(1)

400 FOR I = 1 TO 6
410 X = 60+20*I-(I)3)*60
420 LINE (X,0)-(X,199),4
430 NEXT
440 R$=INPUT$(1)

500 SCREEN 4,6,6
```

On constatera que le programme est assez simplifié par rapport à la première version.

Nous laissons au lecteur le soin d'interprèter chacune des instructions.

Nous ne noterons qu'une différence mineure entre les résultats des deux versions. Dans le premier cas les lignes horizontales Rouge passent "sur" les verticales Bleu. Dans la seconde version ce sont les verticales Bleu qui passent "au-dessus" des horizontales Rouge.

A 2

Programmation de l'Imprimante PR 90 080

L'imprimante à aiguille 90 080 peut imprimer les caractères typographiques classiques (ainsi que quelques autres comme le béta et le mu grecs, les crochets et les accolades), ceci en taille normale (inscrite dans un rectangle de 7 points de haut sur 6 points de large) ou en taille double largeur (inscrite dans un rectangle de 7 points de haut sur 12 points de large).

Elle peut également imprimer des caractères spéciaux, conçus par l'utilisateur.

En particulier il est possible de lui faire tracer des lignes horizontales ou verticales d'épaisseur variées, pour tracer des cadres, des grilles ou souligner des résultats.

Mais rien n'empêche de créer des caractères grecs, russes ou hébreux (dans ce dernier cas un certain traitement de texte doit être mis au point pour imprimer les chaînes de caractères à lire de droite à gauche !).

Nous allons donner ici les méthodes pratiques pour programmer l'imprimante, sans en donner de justification informatique que le lecteur pourra trouver dans le "Mode d'emploi" joint à la machine. Notre but est de donner des méthodes générales facilement applicables et non d'entrer dans le fonctionnement interne des logiciels d'impression.

A/ IMPRESSION O'UN LISTING

L'impression d'un listing est obtenue par la commande directe :

```
LIST "LPRT:(80)"
```

pour obtenir un listing sur 80 colonnes.

Les commandes directes suivantes sont également utilisables pour obtenir des listing partiels :

```
LIST "LPRT:(80)",100
LIST "LPRT:(80)",100-200
LIST "LPRT:(80)",-100
LIST "LPRT:(80)",300-
```

Le résultat est évident. La première commande donne le listing de l'étiquette 100 uniquement, la seconde le listing de l'étiquette 100 à l'étiquette 200, la troisième du début du programme à l'étiquette 100, la quatrième de l'étiquette 300 à la fin du programme.

B/ MISE EN SERVICE DE L'IMPRIMANTE PAR PROGRAMME

L'instruction de mise en service de l'imprimante s'écrit :

```
OPEN "O",1,"LPRT:(8Ø)"
```

précédée, bien entendu, d'un numéro d'étiquette (nous ne repréciserons pas ce détail pour les autres instructions) -(M.Ref 177)-.

Le chiffre 1 est le numéro choisi pour le canal de transmission des données à imprimer. Il est possible de choisir toute autre valeur entre 1 et 16 pour ce numéro de canal, si l'on est amené à utiliser plusieurs canaux en même temps (lecture d'un fichier sur cassette ou sur disquette par exemple).

La lettre O entre guillemets signifie que le canal est utilisé dans le sens mémoire centrale vers périphérique (sens OUT).

Le périphérique est ici l'imprimante caractérisée par le sigle LPRT:(8Ø) placé entre guillemets. On prendra garde à ne pas oublier les deux points ":" après LPRT. La valeur 80 entre parenthèses est le nombre maximal souhaité de caractères par ligne imprimée (on peut fixer un nombre plus faible).

C/ COMMANDES PROGRAMMABLES DE L'IMPRIMANTE

En pratique il est possible de programmer huit commandes d'impression par des instructions de la forme :

```
PRINT #1,XX$
```

(dans la mesure où le canal retenu est le canal 1, sinon faire suivre le signe dièse du numéro de canal choisi).

La variable XX$ est l'une des huit variables suivantes :

a/ MC$=CHR$(15)

Commande la mise en service du Mode Caractères Normaux (caractères typographiques inscrits dans un rectangle de 7 points de haut sur 6 de large).

Cette commande n'est utile que si un autre mode a été précédemment mis en service. Sinon l'imprimante se trouve programmée pour travailler dans ce mode à la mise en route.

b/ MG$=CHR$(8)

Commande la mise en service du Mode Graphique.

Nous verrons plus loin comment bâtir les caractères spéciaux et comment les programmer.

c/ DL$=CHR$(14)

Commande la mise en service du Mode Caractères Doubles.

La programmation des caractères doubles se fait comme celle des caractères simples, en se souvenant toutefois qu'ils occupent une place double, ce dont il faut tenir compte pour les positionner.

d/ PC$=CHR$(16)

Permet de positionner un caractère ou une chaîne de caractères sur une ligne lorsque l'imprimante est en Mode Caractères Normaux ou en mode Caractères Doubles.

Cette commande joue un rôle analogue à celui de l'instruction LOCATE pour l'écran. Nous en verrons l'usage plus loin.

e/ PG$=CHR$(27)+CHR$(16)

Joue un rôle analogue à la commande précédente pour les caractères spéciaux, lorsque l'imprimante est en Mode Graphique. Nous en verrons l'usage plus loin.

f/ NSL$=CHR$(13)

Interdit un saut à la ligne après l'impression d'un caractère, d'une chaîne de caractères ou d'un caractère graphique. Cette commande est utilisable dans tous les modes de fonctionnement.

g/ SL$=CHR$(10)

Impose un saut de ligne après l'impression d'un caractère, d'une chaîne de caractères ou d'un caractère graphique. Cette commande est utilisable dans tous les modes de fonctionnement.

h/ RR$=CHR$(28)

Permet en Mode Graphique de répéter un "motif" composé da 7 points verticaux ("noirs" ou "blancs"). Nous en verrons l'utilisation plus loin.

Il est recommandé, lorsque l'on projette l'utilisation de l'imprimante, de placer en début de programme les huit instructions suivantes sous une même étiquette (les noms mnémotechniques donnés aux variables de commande ont été choisis de façon à en faciliter l'emploi) :

```
10 MC$=CHR$(15):MG$=CHR$(8):DL$=CHR$(14):
   PC$=CHR$(16):PG$=CHR$(27)+PC$:NSL$=CHR$(13):
   SL$=CHR$(10):RR$=CHR$(28)
```

D/ FONCTIONNEMENT EN MODE CARACTERES (NORMAUX OU DOUBLES)

Le positionnement d'un caractère ou du premier caractère d'une chaîne se fait en utilisant la commande PC$ de la façon suivante :

La commande

```
PRINT #1,PC$;"3Ø";ZZ$
```

laisse 30 espaces depuis le début de ligne, avant d'imprimer la chaîne de caractères ZZ$.

Autrement dit, en comptant les colonnes de 0 à 79, la commande PC$;"3Ø" est équivalente à un LOCATE 3Ø,.

On notera, si l'on désire placer ZZ$ en laissant un nombre d'espaces inférieur à 10, que la chaîne, à placer après la commande PC$, doit s'écrire avec un "Ø" comme premier chiffre, par exemple PC$;"Ø5" (l'oubli du zéro provoque des "catastrophes" difficilement prévisibles !).

Si l'on est en Mode Double Largeur la commande est la même, les espaces étant toujours de simple largeur.

Si, par exemple on désire imprimer la chaîne ZZ$="ESSAI D'IMPRESSION" au centre de trois lignes successives en mode normal, en mode double, puis à nouveau en mode normal, on programmera :

```
90 ZZ$="ESSAI D'IMPRESSION"

100 OPEN "O",1,"LPRT:(80)"

110 PRINT #1,PC$;"31";ZZ$:PRINT #1,DL$;
    PC$;"22";ZZ$:PRINT #1,MC$;PC$;"31";ZZ$
```

La chaîne ZZ$ ayant 18 caractères il faut donc placer (80-18)/2=31 espaces avant d'imprimer ZZ$ en mode normal. En mode double la même chaîne occupe la place de 36 caractères. Il faut donc placer (80-36)/2=22 espaces avant d'imprimer ZZ$.

On notera que l'absence de point-virgule après ZZ$, provoque le saut à la ligne.

Il est toutefois possible de regrouper les trois instructions en une seule à condition d'inserrer des commandes SL$ de saut de ligne :

```
110 PRINT #1,PC$;"31";ZZ$;SL$;DL$;PC$;
          "22";ZZ$;SL$;MC$;PC$;"31";ZZ$
```

On prendra garde à respecter scrupuleusement la syntaxe de l'instruction en plaçant une virgule après le numéro de canal et des points-virgules entre les variables chaîne qui suivent.

Il est bien entendu possible d'imprimer des variables numériques par des instructions PRINT ou PRINT USING.

Les quelques exemples suivants montreront la façon de bâtir de telles instructions.

```
10 MC$=CHR$(15):MG$=CHR$(8):DL$=CHR$(14):
   PC$=CHR$(16):PG$=CHR$(27)+PC$:NSL$=CHR$(13):
   SL$=CHR$(10):RR$=CHR$(28)

20 IM$="A(#) ! ####.##     A(#) ! ####.##"

30 N=1235.45

40 FOR I=0 TO 9:A(I)=1050*I+I/15:NEXT

100 OPEN "O",1,"LPRT:(S0)"

110 PRINT#1,PC$;"05";"N = ";N

120 FOR I=0 TO 9 STEP 2

130 PRINT#1,PC$;"20";:
    PRINT#1,USING IM$;I;"=";A(I);I+1;"=";A(I+1)

140 NEXT
```

*** 10
Mise en place des commandes d'impression

*** 20
La chaîne de caractères IM$ servira d'image pour l'instruction PRINT USING en 130.

L'image est constituée de caractères comme A(et) ou d' espaces qui seront imprimés tels quels, de caractères dièse pour positionner les valeurs numériques et du caractère ! destiné à imprimer une chaîne composée d'un seul caractère.

Nous en verrons l'usage à l'étiquette 130.

*** 30 - 40
Ces instructions sont destinées à définir les variables N et A(I) que l'on imprimera par les instructions 110 à 140.

*** 100
Instruction de mise en service de l'imprimante.

*** 110
L'imprimante laisse 5 espaces, imprime la chaîne "N = ", puis la valeur numérique de N. Cette valeur n'est pas "cadrée". Le premier chiffre se place deux espaces après le signe =. Un espace est imposé par la chaîne "N = " et le second représente la place du signe (le signe + n'est pas imprimé).

*** 120 - 130
Boucle en I destinée à l'impression des A(I).

La première instruction de 130 impose 20 espaces en début de ligne.

On notera le point-virgule après la chaîne "2Ø", évitant un saut à la ligne.

La seconde instruction imprime, avec l'image IM$, la valeur numérique de I qui se placera entre les parenthèses de A( ), le signe = (positionné par le symbole ! dans l'image IM$), la valeur numérique de A(I) avec 4 places pour la partie entière et deux décimales, 4 espaces imposés par l'image IM$, la valeur numérique de I+1, entre les parenthèses de A( ), le signe = et enfin la valeur numérique de A(I+1).

On peut s'étonner de voir la façon dont nous avons imprimé les signes = entre les A(I) et les valeurs numériques en traitant le signe = de façon différente des chaînes A(et).

L'expérience montre que l'on peut placer, dans une chaîne image, des constantes chaînes composées de la plupart des caractères alphanumériques -(M.Ref 96)- sauf des caractères utilisés comme descripteurs, soient !, % et & pour positionner les variables chaînes et le dièse pour positionner les valeurs numériques (ainsi que la virgule, l'astérisque, le dollar et la flèche d'exponentiation à certaines places) -(M.Ref 90)-.

Par contre le signe =, figurant à l'intérieur d'une image, n'est pas imprimé ! La seule façon de l'introduire est celle proposée dans l'instruction 130. La même méthode sera utilisée pour introduire les caractères utilisés comme descripteurs.

On notera enfin que l'on peut écrire l'image IM$ de la façon suivante :

```
20 IM$="A(#) ! ####,.##     A(#) ! ####,.##"
```

en inserrant une virgule avant le point décimal -(M.Ref 95)-. Les valeurs numériques A(I) seront alors affichées avec une virgule tous les trois chiffres de la partie entière, séparant ainsi les puissances de 1000. Cette notation anglo-saxonne est à l'opposé de la notation française, mais elle peut être commode pour clarifier les résultats.

E/ FONCTIONNEMENT EN MODE GRAPHIQUE

Lorsque le passage en mode graphique a été commandé par :

```
PRINT #1,MG$;
```

une ligne d'impression doit être considérée comme une succession de 6 x 80 = 480 "caractères" composés chacun de 7 points sur une même verticale, chacun de ces points pouvant être imprimé ou non.

Nous appelerons "motif" un ensemble de 7 points verticaux.

Voyons tout d'abord comment composer un motif. A chaque motif, nous faisons correspondre un nombre bâti de la façon suivante.

Un point non imprimé compte pour zéro. Un point imprimé compte pour 1, s'il est sur la ligne supérieure, pour 2, s'il est sur la seconde, pour 4, s'il est sur la troisième, pour 8, s'il est sur la quatrième, pour 16, s'il est sur la cinquième, pour 32, s'il est sur la sixième et enfin pour 64, s'il est sur la ligne inférieure.

On fait la somme des nombres attibués à chaque point et l'on ajoute 128. Le nombre N ainsi obtenu est alors transformé en un caractère par la fonction CHR$(N).

Un caractère spécial composé de P motifs différents successifs est alors mis en mémoire en concaténant les P caractères CHR$(N1),...,CHR$(NP).

Précisons ces notions par un exemple. Nous voulons bâtir le caractère spécial suivant :

```
OOOOOOOOOO..        1
O........O..        2
O..OOOO..O..        4
O..O..O..O..        8
O..O..OOOO..       16
O..O........       32
O..OOOOOOOOO       64
```

Nous avons rappelé, à droite du schéma, le "poids" de chaque point.

Le premier motif sera représenté par

N1 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255.

Le second et le troisième motifs seront représentés par

N2 = N3 = 1 + 128 = 129

Le quatrième motif sera représenté par

N4 = 1 + 4 + 8 + 16 + 32 + 64 = 253

Le cinquième motif sera représenté par

N5 = 1 + 4 + 64 + 128 = 197

etc.

La mise en mémoire du caractère se fera donc par la suite d'instructions suivante :

```
20 DATA 255,129,129,253,197,197,221,209,
        209,223,192,192

30 ZZ$="":FOR K=1 TO 12:READ Z:
   ZZ$=ZZ$+CHR$(Z):NEXT
```

Les DATA en 20 sont les 12 nombres caractéristiques des 12 motifs constituant le caractère. Les instructions sous l'étiquette 30 mettent le caractère spécial en mémoire dans ZZ$.

Lorsque le même motif se répète plusieurs fois de suite, il est possible de racourcir le stockage en mémoire en utilisant la commande de répétition RR$ = CHR$(28).

Un exemple va nous faire comprendre la méthode.

Supposons que nous voulions tracer un caractère de soulignage continu composé de deux lignes, l'une placée à la hauteur du second point du motif, la seconde à la hauteur du quatrième point du motif.

Le motif est alors caractérisé par le nombre 2 + 8 + 128 = 138 et donc représenté par ZM$ = CHR$(138).

Si l'on veut souligner 18 caractères typographiques (affichés préalablement en Mode Caractère), il nous faut bâtir un caractère spécial ZSL$ répétant 6 x 18 = 108 fois le motif ZM$, puisqu'un caractère typographique occupe six points. Le caractère ZSL$ est alors construit par

```
ZSL$=RR$+CHR$(108)+ZM$
```

D'une façon générale, pour répéter Q fois le motif ZM$, on bâtit la variable chaîne de caractères RR$+CHR$(Q)+ZM$. La valeur Q ne peut dépasser 255. Avec Q = 0 on obtient la répétition maximale de 256 motifs.

Si nous voulons tracer notre caractère de soulignage sur les 80 colonnes disponibles de la ligne, il nous faut alors bâtir la variable

```
ZTL$=RR$+CHR$(0)+ZM$+RR$+CHR$(224)+ZM$
```

ce qui va bien nous répéter 256 + 224 = 480 fois le motif.

Il nous reste maintenant à positionner les caractères spéciaux.

Il faut pour cela utiliser la commande PG$, c'est-à-dire CHR$(27)+CHR$(16).

En numérotant les points de la ligne de 0 à 479, un caractère spécial sera placé au point N, par l'instruction :

```
PRINT#1,PG$;CHR$(N1);CHR$(N2)
```

avec la règle suivante pour calculer N1 et N2 :

- si N est inférieur à 256 alors N1 = 0 et N2 = N

- si N est supérieur ou égal à 256 alors N1 = 1 et N2 = N - 256

Il faut noter qu'en l'absence de commande de positionnement un caractère est placé en début de ligne. En particulier si l'on tente d'imprimer une suite de caractères spéciaux par une instruction de la forme :

```
PRINT#1,Z1$;Z2$;Z3$;.....;ZN$
```

on constatera que les caractères s'impriment bien les uns derrière les autres, mais que la suite des instructions

```
PRINT#1,Z1$;:PRINT#1,Z2$;:PRINT#1,Z3$;:....:PRINT#1,ZN$;
```

même avec un point-virgule après les variables Z$, conduit à imprimer les Z$, en colonne 0, les uns au-dessous des autres.

Si l'on évite le saut à la ligne en utilisant la commande NSL$ par :

```
PRINT#1,Z1$;NSL$;:PRINT#1,Z2$;NSL$;:
PRINT#1,Z3$;NSL$;:....:PRINT#1,ZN$;NSL$
```

on obtient la superposition de tous les caractères en début de ligne.

Il est donc nécessaire, chaque fois que l'on effectue une impression avec une instruction commençant par PRINT, de positionner le caractère qui suit par la commande PG$.

Enfin une dernière règle pratique (due à la capacité de 90 octets de la mémoire tampon), ne jamais placer derrière une commande PRINT une succession de chaînes de caractères de longueur totale supérieure à 90.

Le court programme de démonstration suivant va préciser l'emploi de toutes ces commandes et règles.

```
10 MC$=CHR$(15):MG$=CHR$(8):DL$=CHR$(14):
   PC$=CHR$(16):PG$=CHR$(27)+PC$:NSL$=CHR$(13):
   SL$=CHR$(10):RR$=CHR$(28)

20 DATA 255,129,129,253,197,197,221,209,
        209,223,192,192

30 ZZ$="":FOR K=1 TO 12:READ Z:
   ZZ$=ZZ$+CHR$(Z):NEXT

40 ZGC$="":FOR K=1 TO 5:ZGC$=ZGC$+ZG$:NEXT

50 ZL$=RR$+CHR$(108)+CHR$(138)

60 ZTL$=RR$+CHR$(0)+RR$+CHR$(224)+CHR$(138)

70 T$="ESSAI D'IMPRESSION"

100 OPEN "O",1,"LPRT:(80)"

110 PRINT#1,PC$;"31";T$

120 P=31:GOSUB 1000

130 PRINT#1,MG$;PGR$;ZL$

140 S=5:GOSUB 2000:PRINT#1,ZTL$:GOSUB 2000

150 FOR K=0 TO 7

160 P=10*K:GOSUB 1000:PRINT#1,PGR$;ZGC$;NSL$;

170 NEXT

180 PRINT#1,MC$

190 S=3:GOSUB 2000

200 CLOSE:END
```

```
1000 N=6*P:IF N)255 THEN N1=1:
     N2=N-256 ELSE N1=0:N2=N

1010 PGR$=PG$+CHR$(N1)+CHR$(N2):RETURN


2000 FOR K$=1 TO S:PRINT#1:NEXT K$:RETURN
```

*** 10 - 30
Nous avons déjà vu ces trois instructions plus haut.

*** 40
La variable ZGC$ à la fin des cinq boucles en K, contient 5 caractères spéciaux définis en 20 et 30, c'est-à-dire 5 "grecques".

Un caractère ZG$ a une longueur de 12 caractères (il est constitué par la concaténation de 12 CHR$). La variable chaîne ZGC$ a donc une longueur de 5 x 12 = 60 caractères, ce qui est inférieur à la valeur limite de 90.

Il ne serait pas possible par contre de bâtir une variable contenant 10 "grecques", car cette variable aurait une longueur de 120 caractères, incompatible avec la capacité de 90 de la mémoire tampon.

Nous avons retenu 5 "grecques", car la figure occupe une place égale à 10 caractères typographiques (60 points). Pour remplir toute la ligne, il suffira d'en imprimer 8 bout à bout.

*** 50
Le caractère spécial ZL$ est composé de 108 motifs -second et quatrième points- (138=128+2+8). Il constitue donc un caractère de soulignage avec deux traits, d'une longueur de 108 points, c'est-à-dire de 18 caractères typographiques.

*** 60
Le caractère spécial ZTL$ est constitué de 480 motifs (256+224), identiques au motif précédent. Il constitue donc un caractère occupant la totalité de la ligne.

*** 70
Définition de la variable chaîne de caractères T$. On notera que T$ est une chaîne de 18 caractères.

*** 100
Instruction de mise en service de l'imprimante.

*** 110
Impression de 31 espaces, puis de la chaîne de caractères T$.

*** 120
On cherche à placer le caractère de soulignage ZL$, sous la chaîne T$ que l'on vient d'imprimer. Il faut donc placer ZL$ en laissant 31 espaces de 6 points. La variable N est donc posée égale à 31 et le sous-programme 1000 va calculer la commande de positionnement en conséquence.

Etudions donc maintenant le sous-programme 1000.

*** 1000
a/ Le nombre N de points correspondant à P espaces est évidemment égal à 6*P.

b/ Calcul des deux valeurs N1 et N2 correspondant à N, suivant la formule indiquée ci-dessus.

*** 1010
a/ La variable de positionnement PGR$ est le résultat de la concaténation de la commande PG$ et des deux caractères CHR$(N1) et CHR$(N2).

b/ Instruction de clôture de sous-programme.

Revenons maintenant au programme principal.

*** 130
L'impression de la commande MG$ provoque le passage au Mode Graphique.La variable de positionnement PGR$, calculée par le sous-programme 1000 en fonction de P, place le caractère ZL$ sous la chaîne T$.

*** 140
a/ et b/ Le sous-programme 2000 provoque S = 5 sauts à la ligne.

c/ Impression de ZTL$ (trait double de soulignage) sur toute la largeur de la page. L'absence de commande de positionnement place le début de ZTL$ au point 0.

d/ Provoque 5 sauts à la ligne.

*** 150 - 170
Une boucle en K, variant de 0 à 7, imprime 8 fois le groupe ZGC$ de cinq "grecques".

On notera que l'instruction PRINT est suivie de la commande de positionnement PGR$ (calculée par le sous-programme 1000 avec P = 10 * K) et se termine par la commande NSL$, suivie d'un point-virgule, pour éviter le saut à la ligne. On obtient ainsi une "grecque" occupant la totalité de la ligne.

*** 180
Cette instruction impose le retour au Mode Caractère.

Elle est indispensable si l'on veut retrouver l'imprimante capable de travailler dans ce mode pour le programme suivant ou capable d'éditer un listing.

Son oubli interdit un fonctionnement autre qu'en Mode Graphique (le dernier commandé).

Un retour au fonctionnement normal peut être obtenu simplement en coupant puis en rétablissant l'alimentation par l'interrupteur de l'imprimante.

*** 190
Provoque 3 sauts à la ligne pour dégager les impressions effectuées.

*** 200
a/ Provoque la fermeture de tous les canaux ouverts.

Cette instruction est indispensable pour permettre une utilisation ultérieure de l'imprimante.

Son oubli entrainerait, sur commande de listing, un message d'erreur IU (Périphérique en cours de fonctionnement, "device In Use").

Le retour à un fonctionnement normal se fait simplement en tapant directement au clavier la commande CLOSE.

b/ Instruction nécessaire pour éviter au programme de pénétrer en séquence dans le sous-programme 1000, provoquant le message d'erreur RG (RETURN without GOSUB).

On notera que cette instruction provoque automatiquement la fermeture de tous les canaux, rendant de ce fait inutile l'instruction CLOSE précédente.

Nous avons néanmoins laissé l'instruction CLOSE pour en montrer la nécessité (certains programmes peuvent ne pas se terminer par une instruction END).

A 3

Index des instructions

Cet index donne, pour chaque instruction, le numéro du chapitre et le numéro de l'étiquette où elle a été utilisée. Bien entendu, tous les programmes et toutes le étiquettes ne sont pas cités. Seuls les cas les plus typiques ont été retenus.

On remarquera enfin, que toutes les instructions figurant dans le manuel de référence ne figurent pas dans cet index. Seules, les instructions les plus courantes ont été utilisées dans les programmes de démonstration de cet ouvrage.

ASC 1/80 5/2080 6/320 7/60030

ATTRB 4/200 A1/50

BEEP 5/180 10/6020

BOX 4/30 6/20

BOXF 2/100 4/40 8/110 A1/30 A1/40

CHR$ 1/20 2/40 6/280 9/60

CLEAR 5/10 6/10 7/10 8/10 9/20 10/60 10/10

CLOSE 9/4130 10/6570

CLS 1/10

COLOR 1/60 8/300 8/800

CONSOLE 4/520 5/2000 5/2100

COS 4/120

DATA 2/10 6/200 7/20 8/30 11/40 11/60 11/80

DEFDBL 8/10

DEFGR$ 5/40 5/270 6/150 7/240

DEFINT 5/20 9/30

DIM 5/20 6/250 8/10 10/80

END 5/4000 6/320 8/5020

FOR NEXT 1/30

FRE 5/1555

GOSUB RETURN 3/210 7/540

GOTO 2/610

HEX$ 7/Page 146

IF GOTO ELSE 1/80 2/510 2/520 5/210

IF THEN ELSE 3/240 3/270 3/300 5/2020 6/320 6/340 8/2520 11/530 11/1090 11/2500

INKEY$ 2/500

INPUT# 7/520 7/530

INPUT 10/40 10/130

INPUT$(N) 1/80 2/10000 6/310

INPUTPEN 5/200 8/900 8/5000

INT 5/240

LEFT$ 5/1560

LEN 5/1560 8/1110 9/2000 11/5000 11/1000

LINE 2/70 4/70 6/2000 A1/90

LINEINPUT 5/530 5/550

LIST A2/Page 274

LOADM 7/30 10/40 10/20

LOCATE 1/60

MID$ 3/1020 5/1540 7/60030

NEXT 1/70

ON GOTO 2/540 5/1010 5/3020 6/390 9/510

ONPEN GOTO 2/600

OPEN 5/3500 9/4000 10/6500 10/30 A2/100

PEEK 7/220 7/60040 10/70 10/140

PEN 2/40

PLAY 2/10010 8/930

POKE 7/220 10/50 10/6550

PRINT 2/40 2/170 5/260

PRINT# 5/5000 9/4010 10/6510 A2/110

PRINT# USING A2/130

PRINT USING 1/60 3/80 5/30 5/2050 9/1020

PSET A1/20 A1/50

PTRIG 2/510 5/210

READ 2/120 6/260 7/210 8/40 8/50
9/220 11/50 11/70 11/90

RESTORE 2/10000 6/270 9/4030

RETURN 3/1040 8/5050

RIGHT$ 5/3530

RND 5/1220

SAVEM 7/300 10/6570 10/130

SCREEN 1/10 8/100 A1/10 A1/190

SCREENPRINT 1/90

SIN 4/120

STEP 5/50 5/90 9/150

STR$ 5/1540 5/3530 8/2500

VAL 8/3100 8/3400 9/2100 11/5020

Achevé d'imprimer
par **Cid** *éditions*

Dépôt légal : janvier 1984
N° d'Éditeur : 4046